Minds, Machines and Mathematics

Artificial intelligence

Everything is mathematical

Minds, Machines and Mathematics

Minds, Machines and Mathematics

Artificial intelligence

Ignasi Belda

Everything is mathematical

© 2011, Ignasi Belda (text)
© 2013, RBA Contenidos Editoriales y Audiovisuales, S.A.U.
Published by RBA Coleccionables, S.A.
c/o Hothouse Developments Ltd
91 Brick Lane, London, E1 6QL

Localisation: Windmill Books Ltd.
Photography: iStockphoto, Corbis, Getty Images

All rights reserved. No part of this publication can be
reproduced, sold or transmitted by any means without
permission of the publisher.

ISSN: 2050-649X

Printed in Spain

Contents

Preface 7

Introduction 9

Chapter 1. What is Artificial Intelligence? 13
The Turing test 13
Research 19
Learning 22
Planning 23
Automated reasoning 25
Natural language processing 31
Managing what we know 33

Chapter 2. Research 37
Darwin said it first 38
Initialisation 39
Evaluation 41
Selection 42
Reproduction 46
Replacement 47
A practical example: evolving a good drug 50

Chapter 3. Machine Learning 55
An example of learning: predicting cancers 56
Another example: online marketing 60
The robot's brain: neural networks 64
Neurons cluster together 66
And so the brain functions 71
The brain grows ever more complicated 73
Are exams necessary? 75

Chapter 4. Automated Planning and Reasoning 81
How a transplant is managed 81

Planning is the operative word 85
Conflict detection 90

Chapter 5. Data Analysis 95
Data mining 96
The curse of dimensionality 99
Data visualisation 103
Pattern recognition 106
A practical example: sales analysis 107

Chapter 6. Artificial Life 111
An introduction to artificial life 111
Complex adaptive systems 114
    First property: aggregation 115
    First mechanism: tagging 115
    Second property: non-linearity 117
    Third property: flow formation 118
    Fourth property: diversity 121
    Second mechanism: internal models 123
    Third mechanism: building blocks 124
Cellular automata 125
Artificial immune systems 127
Swarm intelligence 131
Applications of artificial life 132
    Game theory 133
Back to data mining 134
Robot programming 136

Epilogue 139

Appendix: Conversation with Eliza 141

Glossary 143

Bibliography 147

Index 149

Preface

I do not fear computers.
I fear the lack of them.
Isaac Asimov

For several decades, artificial intelligence has excited many, and here we will try to separate the fact from the fiction in the films, TV series and novels that often feature talking robots, autonomous machines and other automated systems capable of behaving like real human beings.

In this book we will try to get to the bottom of some of the mysteries associated with the two words 'artificial intelligence'. Having done that, we will also briefly introduce the concept of 'artificial life'. So, will the near future be inhabited by talking machines? And in the medium term? How far are we from being able to interact with autonomous, intelligent systems that can draw energy from food just like a living being? Is this all a fiction?

We will examine all these questions and many more during the course of the book. More specifically, we will analyse artificial intelligence in its four main applications: search, learning, planning and automated reasoning. In Chapter 5, we will also discuss data analysis, one of the fields in which smart tools are most widely used, and one which is hugely important to our digital era. Billions of pieces of data are now generated every second and would be useless without the smart tools needed to extract intelligence from them.

Finally, in Chapter 6 we will take a quick plunge into the depths of artificial life. What is a living being? And what is an 'artificial' being? Will we one day have the ability to create living, intelligent organisms with which we can interact? Are we perhaps already capable of doing it? All these questions will be answered using practical everyday examples, in which we might recognise the 'artificial living' beings that we already meet in our daily lives.

To end this preface, it just remains for us to point out that although computer science is a discipline of applied mathematics, we have largely strived to 'conceal' the more mathematical and analytical side that underpins all the techniques and examples given, to give readers a general overview of this interesting topic. And we seem to have succeeded, to the extent that on more than one occasion our editor, an acclaimed and experienced champion of mathematics, asked us: is this book really about mathematics? It certainly is and, in fact, without mathematics, nothing we present here would be possible.

17 November 2067

Great Central Computer's methods spark controversy

Major unrest in European capitals due to protests against the government's welfare cuts

Paris, Brussels, Barcelona, Milan, Athens, Munich and many other European cities have been hit by a wave of protests and popular uprisings against the latest measures approved by Great Central Computer (GCC).

The new legislation will have a direct impact on the large European middle class, as the set of approved measures cut the number of days of annual holiday by 10%, from 200 to 180, reduce cabin temperatures by 1°C, from 25 to 24°C, and limit benefits to the first four personal robot assistants, meaning a 20% reduction in the welfare budget.

This set of measures will make savings worth £8,000 billion, which will be used o stimulate mining productivity in the colonies of Mars and Venus, according to GCC, which has governed the European System with an iron first for the last 34 years. As is widely known, the European Constitution, revised 39 years ago and ratified by the People, grants sweeping executive, legislative and judicial powers to GCC, whose computing capacity, memory and analytical speeds far exceed the capacities of any human.

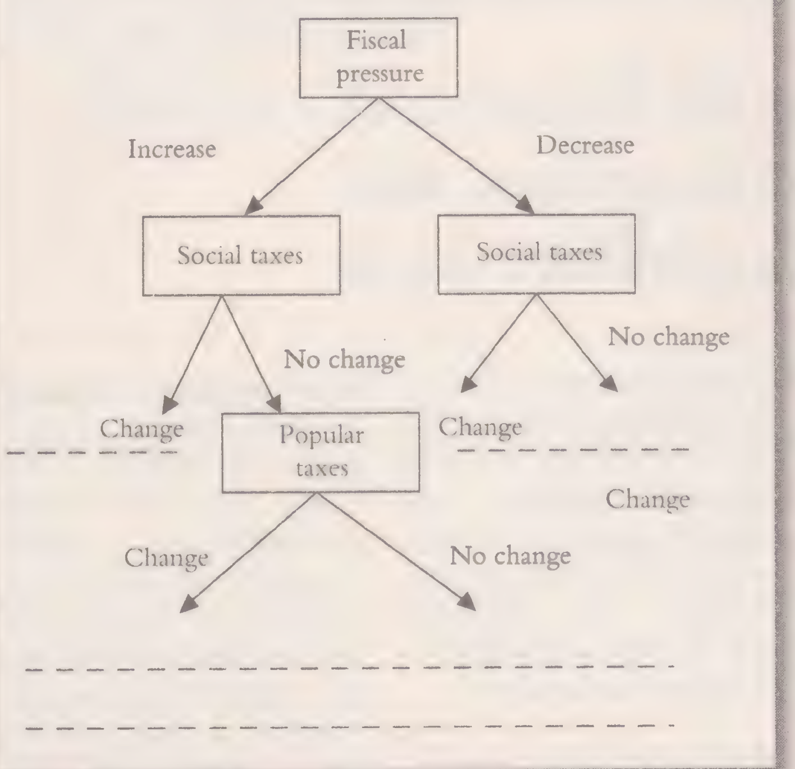
Although various Automatic Analysts (AA) from the main Independent Automatic Analysis Systems (IAAS) have corroborated the effectiveness of GCC's package of measures, representatives of the People claim it is a direct attack on citizens' freedoms in response to the human decision not to approve a Universal Declaration of Rights of Robots and Autonomous Machines.

In the bowels of Great Central Computer

GCC is guided by a cognitive 'map' with trillions of variables, each contained in what are known as 'neurons' which, at the time of construction, were connected to the neighbouring neurons, creating a huge neural network. This neural network evolves, which means that after each new event, the value of the corresponding variable changes; in a kind of chain reaction, the neurons' connections then also change. To explain it graphically, imagine a stone thrown into a pool. The point where the stone hits the surface of the water is subject to a change in surface tension, which is transmitted almost instantaneously to the other water molecules on the surface of the pool, thereby creating the familiar waves which ripple across the water until it calms down.

When GCC was built, a quadrillion data items covering the whole of human history were uploaded to its cognitive map. The map then self-constructed its neural pathways based on past experience.

Since it built its network of neural pathways, GCC has used it not only to keep the



Tree diagram of political measures

European System in check, but also to make decisions on how to act, which it does using an automated reasoning system. This system is capable of making hypotheses and predicting their effects. Going back to our analogy, it throws the stone into the pool, but a virtual pool rather than a real one, and observes how the surface of the water changes. If the effect is positive, it executes the change.

Simplifying greatly, GCC's automated reasoning system is a search algorithm. If GCC detects a problem that needs solving or a shortage of any kind, it deploys a tree of combinations that reflect all the political, social, economic and even military options that could be taken. As politics is such a complex process, within milliseconds this tree already contains millions of branches, and if nothing else were done, within seconds it would have more branches than there are atoms in the Universe.

Suppose the aim is to cut the fiscal deficit by 1.5%. At the computing speed of modern quantum supercomputers, a tree diagram like this one becomes infinitely large within seconds of initiating the process. As it is obviously impossible to compute an infinite tree, GCC uses a range of tools to 'prune' some of the branches of the tree where it sees they do not present a viable path to the desired goal. For example, if we want to reduce the fiscal deficit without altering any taxes or stimulating growth in the economy via a fiscal expansion, it can immediately predict that this path leads to a dead end. Therefore, all those branches of the tree that correspond to this type of solution can be discarded.

These 'problem-solving' tools, known as *heuristics*, were built automatically using historic data. After the event, a team of social science researchers from all over Europe carried out a painstaking review of them. To the surprise of the sceptics, the modifications made by the experts to the first versions of the heuristics accounted for just 0.003% of the total. The expert review took five years to complete, while the automated reasoning process had taken just three days.

This automated reasoning system using heuristics is based on scalable algorithms, in other words 'intelligent systems' which provide random solutions (in this case, each solution is a suggested heuristic), and continues to refine them over time, attempting to simulate a process of natural evolution underpinned entirely by the laws of evolution laid out by Darwin. This means that the solutions best suited to the environment are those that have the most descendants, or in other words, the heuristics which find the most support in the historic data are most likely to be propagated in this process of virtual evolution and therefore have descendants.

Now all that remains to be seen is whether the measures suggested by Hercules v3.4 set our society on the right track, as has been the case over the last 34 years, and whether it all ultimately delivers a substantial improvement in our quality of life.

As readers may have noticed, this news story is a complete fiction and far removed from anything possible today. But might a scenario like this be possible within 50 years? A scenario in which the major decisions that determine the destiny of humankind are taken, controlled, monitored and analysed by thinking machines?

In fact, as we will see in Chapter 4, in Cyprus, where the political and military situation is extremely complex, researchers from the local university and the Bank of Greece have already proposed a system that uses cognitive maps to predict the stability of the system in response to changes considered by any of the actors involved: Greece, Turkey, NATO, the EU, etc.

To get a clearer idea of how plausible the first story is (or isn't), we will assess the current state of artificial intelligence so that we can try to determine how remote the possibility of such a scenario is. Welcome to the exciting world of artificial intelligence, in which mathematics, computing and philosophy come together and push the very limits of what makes us human.

Chapter 1

What is Artificial Intelligence?

Science fiction films shown on television often portray autonomous machines capable of making their own decisions. Where is the line dividing fact from fiction in these films? How much progress has artificial intelligence made? Will we soon have reached the stage where we can develop systems like those in *2001: A Space Odyssey* or, more recently, the film version of *I, Robot*?

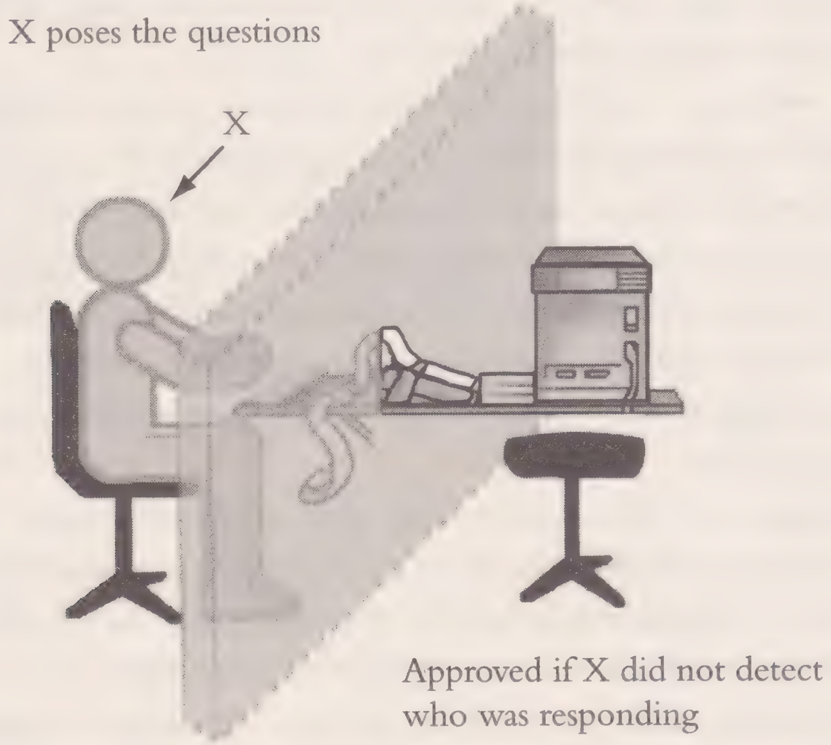
Before we begin, however, we should give a precise definition of the concept we're exploring. By 'artificial' we can certainly all agree that we mean things that are 'not natural' or 'made and created by human will'. But, what is 'intelligence'? According to most dictionaries, the word 'intelligence' has many accepted meanings, including the 'ability to understand or comprehend', 'ability to solve problems' and 'skill, dexterity and experience'. In reality, the fact that such widely different definitions of this word exist highlights the complexity behind the concept.

Psychologists and philosophers have tried to delimit, define and measure intelligence over the course of the centuries. But these metrics are even more vague when applied to non-human entities. For example, would we say that a computer program that can synchronise and coordinate a complex flight motor system which has the autonomous ability to modify an air route according to current needs and has a 100% reliability record is intelligent? Yes, we probably would. On the other hand, would we say the same of a mosquito? This insect is capable of coordinating a complex flight motor system, makes autonomous decisions on air routes and boasts 100% reliability in its air operations.

The Turing test

The first practical solution to the question of whether a given non-human entity is intelligent was suggested in 1950 by mathematician Alan Turing, regarded as one of the fathers of artificial intelligence. He created the Turing test, which determines the existence of intelligence in a machine on the basis of a very simple idea: if a machine behaves in all respects like an intelligent being, then it must be an intelligent being.

The test involves putting a human assessor and the machine being assessed in two different rooms separated by a partition preventing one from seeing the other. Next, using a keyboard and screen, the assessor puts a series of questions to the entity being assessed, which answers them. If the assessor thinks that the entity answering the questions is a human being, it is deduced that the machine being assessed is intelligent, and therefore has artificial intelligence.


X poses the questions
X
Approved if X did not detect who was responding

The person who runs the test cannot see the machine (X). In this way, it is only by analysing the answers that the assessor can decide whether or not it is a human being.

Inevitably, the Turing test was on the receiving end of a barrage of criticism from some theorists. Can we say that a machine is intelligent just because it answers the questions using an enormous dictionary of questions and answers? Or is intelligence something more than an apparent form of behaviour? Is it, for example, the existence of consciousness?

ALAN TURING (1912–1954)

The British mathematician Alan Turing is regarded as one of the fathers not only of artificial intelligence, but also of modern computing, because of all the fundamental theoretical contributions he made to the science during the 42 years of his life. During the Second World War, he worked as a cryptographer for the British armed forces, and was one of the main players in breaking the Enigma machine codes, thanks to which the Allies were able to anticipate the movements of Nazi forces.

His main theoretical contribution to computer science was what is now known as the 'Turing machine', a theoretical model of a universal computer. A universal computer is one capable of processing any input data and delivering its output data in a finite period of time. The Turing machine consists of an infinite tape with symbols written on it, a head that can move left or right on the tape, read the symbols, erase them and write new ones, and rules that determine the behaviour of the head according to each possible symbol that it detects on the tape. In practical computing, these rules would symbolise a computer program, and the tape would represent the program's input/output system and a register of its execution status.

Nowadays, when a new programming language is devised, like C, Pascal, Java, etc., the first thing that has to be formally demonstrated is that the new language is Turing-compatible, i.e. equivalent to a Turing machine.

Unfortunately, Turing took his own life because of the persecution he suffered at the hands of the British legal system for his homosexuality. At his trial, he decided not to defend himself because he didn't think he had anything to apologise for. Consequently, he was found guilty, but Turing chose to be chemically castrated rather than sent to prison. This caused severe physical changes which are thought to have led him to commit suicide. In 2009, British prime minister Gordon Brown issued an official apology for the treatment Turing received during the last years of his life.

The main criticism of the Turing test was presented by the philosopher John Searle in the Chinese room experiment. Imagine that any given group of people who do not speak a word of Chinese are placed in a closed room in a Shanghai shopping centre. Next, passers-by are requested to ask the people in the room questions. To do so they have to write them down and post them through a slot. On the other side of the wall, inside the room, there is a manual which clearly lists all the characters in the Chinese alphabet that make up the answers to every one of the questions that might be asked. Take an example: a young man writes

CAN INTELLIGENCE BE FAKED? CHESS, KASPAROV AND DEEP BLUE

Chess is a classic combinatorial problem to which smart techniques have been applied, without success, since the early days of computing with the aim of beating human players. So, why can't intelligence be faked in games like chess? Imagine we program a computer with the rules of the game, and the computer compiles the list of all the possible strategies. Then we can note down the best move to make, strategy by strategy. However, the number of possible strategies is in the order of 10^{123}. This number is larger than the total number of electrons in the Universe! We would need a memory larger than the total mass of the Universe just to store the results! So in the case of chess, unlike the Chinese room, it is totally impossible to fake intelligence using a glossary of plays and moves.

The most controversial example of a man-versus-machine chess match was the battle between Deep Blue and Garry Kasparov. In 1996, Deep Blue became the first chess supercomputer to beat a human world champion. However, the aggregate score in all their matches was 4-2 in favour of the Russian grandmaster. At that time, Deep Blue could analyse 100 million moves per second. The controversy arose when Kasparov played a second version of the computer, Deeper Blue, which could analyse 200 million moves per second. On this occasion the machine won, but Kasparov complained that at one point during the match the computer had had the help of a human operator. Kasparov laid a trap where by sacrificing a pawn, he would be able to launch a counter-attack in subsequent moves. This trap was impossible for the computer to detect, because its analytical capabilities were limited to a certain number of future moves, before the point when Kasparov would launch his counter-attack. However, the computer did not fall into the trap, raising the Russian champion's suspicions. Kasparov later asked for the written logs of the machine's processes; IBM agreed to provide them, but never did.

in Chinese on a piece of paper, "Is it warm in there?" and puts it through the letterbox. Next, the people inside the room look at the Chinese characters, look them up in the manual and select a possible answer to the question. They then transcribe it, dash by dash, on a blank piece of paper and hand it back through the slot to the young man who asked the question. On the sheet of paper is written, in Chinese: "No, in fact it's freezing". Both this young man and the other assessors are being given coherent responses in their own language, so naturally they will think that the people in the room can speak fluent Chinese. Yet the individuals

Deep Blue, IBM's supercomputer, which beat the world chess champion.

in the room based their answers on a code book and did not understand a single word of the conversations.

But, can we suspect that a machine that passes the Turing test might be tricking us in a similar way? The answer is no. The Chinese room is a deceptive scenario because in reality, while it's true that the people in the room don't speak Chinese, the person who did answer the questions was an amalgam of entities formed by the people and the manual. Although we can't say that the manual 'speaks' Chinese, what we can say is that it was put together by somebody who did, because otherwise he or she wouldn't have been able to write the set of questions and answers.

In practical terms, a new technology is now regarded as intelligent if it is capable of solving a problem creatively, something that has always been regarded as the exclusive domain of the human brain. A representative example of technology that seems intelligent but is not regarded as such would be the first 'expert systems' that appeared in the 1960s. An expert system is a computer program that has been programmed with rules of varying complexity, and which can autonomously control some systems. An example might be a computer programmed with a huge list of medical symptoms so that, given a new patient and his symptoms, it can decide what treatment the patient needs. However, if the system is not able to create a new rule deduced from previous rules or invent a new treatment when the situation requires it, it is not regarded as being creative and, consequently, it is not intelligent.

So for a computer program to be considered intelligent a number of rather subjective criteria need to be met, such as having the capacity to learn complex subjects, to optimise mathematical functions with many parameters (dimensions) and within a huge interval (scope), or to plan a large quantity of resources with defined constraints.

As in other areas of science and technology, artificial intelligence has specialised and divided into five main branches:

1. Research.
2. Learning.
3. Planning.
4. Automated reasoning.
5. Natural language processing.

However, the technologies and algorithms used in the various specialised areas are the same in many cases. Next we will summarise these branches and give some practical examples.

Research

Research refers to the process of searching for the optimal solution to a given problem. When this problem can be defined in terms of a mathematical function we are in the realm of function optimisation, i.e. the search for input parameters that maximise the output of the function. Problems often involve the simultaneous optimisation of various functions, and these functions are also difficult to define and delimit. For an automated system, function optimisation is a complex problem, especially if there is no analytical formula for the function and the 'form' of the function can only be inferred from a few data points. Similarly, in many cases the function in question has hundreds of separate parameters to be adjusted, several hours of computation are required to obtain each data point or the data contain noise, meaning the value of the function at a given point in space is not exact.

Artificial intelligence is used to tackle these complex scenarios. It's worth noting that a human being can solve complex multidimensional mathematical functions instinctively in seconds. Similarity functions are good examples of this. Imagine you know more than 500 people, but if you see a photograph of one of them you will quickly be able to tell whether it's a photo of one of your acquaintances, and which one of them it is. This seemingly simple operation is solved mentally by optimising a function that measures the differences between the faces you keep stored in your memory and the face in the photograph you're looking at. A face contains thousands of observable characteristics or dimensions, such as eye colour, the size relationship between the mouth and nose, whether there are freckles, etc. Our brains are capable of detecting all these characteristics and comparing them with the faces of all the people we know. They can measure the distances in the faces in the photograph and compare them with all the others, and find the face in which the distance is smallest. They can also decide whether or not this distance is small enough to deduce that the person in the photo and the person in your memory are one and the same. The human brain performs all these operations in less than a second. However, for a computer, face recognition is a highly complex operation; a modern computer would probably take minutes to find the solution.

GO, ONE OF THE GREAT UNRESOLVED CHALLENGES OF ARTIFICIAL INTELLIGENCE

The game Go is a good example of a combinatorial problem in which a semi-trained human can detect at a glance the smartest strategy for any scenario, a task which a computer finds extremely difficult. Thus far, no computer program has been able to beat any professional player without imposing initial handicaps on the latter.

This Chinese strategy game has extremely simple rules that give rise to scenarios of great strategic complexity. It involves the use of a board with a 19 × 19-line grid on which the players take turns to place white and black pieces at vacant intersections. If a piece or group of pieces is completely surrounded by pieces of the opposing colour, the group is captured and the pieces are removed from the board. A player can miss her go if she thinks it is advantageous to do so, but if both players pass in consecutive turns the game ends and the winner is the player who at that time dominates a larger section of the board.

Mathematically, Go is classified as a game of strategy just like chess. However, while there are computer programs capable of beating world chess champions, Go programs find it hard to beat experienced players. This mainly happens for three reasons.

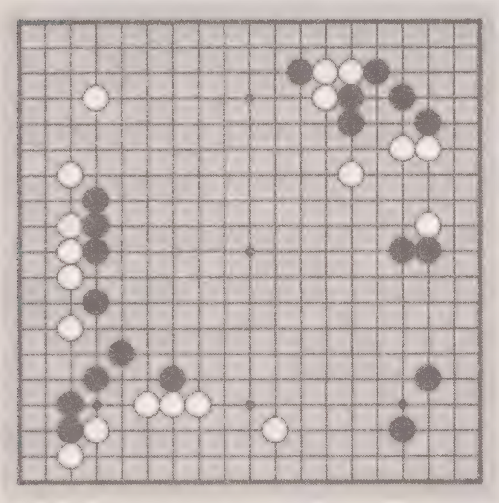
First, because of the dimensions of the Go board, which is more than five times larger than a chess board, so a higher number of possible moves need to be analysed.

Second, because a move in Go can affect hundreds of subsequent turns, making it impossible for a computer to make such long-term predictions.

And finally, in chess the pieces are captured one by one and all have a set value, meaning fairly precise assessments can be made of the benefits of a move.

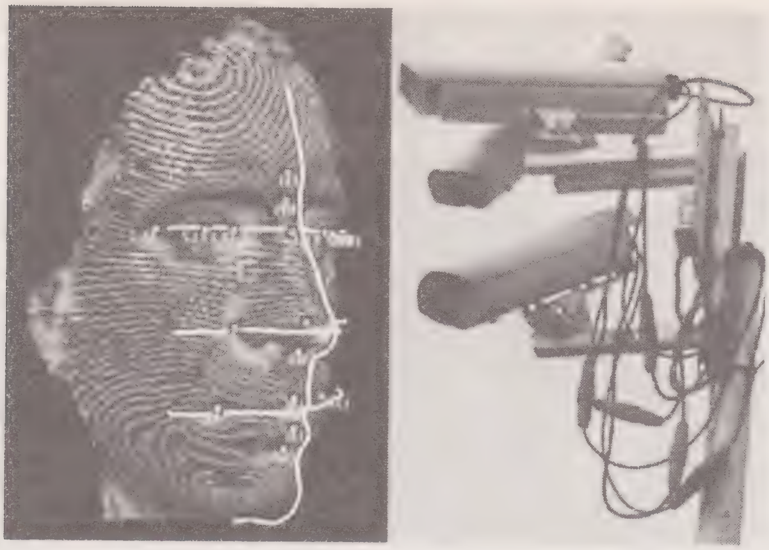
However, in Go the benefits of a capture depend entirely on the pieces captured, and these benefits depend on the position of the pieces at the time.

But why do we call this branch of artificial intelligence 'research' if we are talking about numerical optimisation? Research includes solving other types of problems, including 'combinatorial' problems. A combinatorial problem is one where the solution is formed of distinct elements that can be combined to produce a combinatorial space. The solution is determined by the optimum set of elements. A good example of a combinatorial problem is a game of chess, in which the optimum



Top, Go board and pieces, called 'stones'. Left, game situation in the final of the 2002 World Championships between Choe Myeong-hun (white) and Lee Sedol at the end of the opening phase.

solution is a series of piece moves that deliver a win. Another classic example is the famous rucksack problem, in which various items can be placed in a rucksack prior to a trip. The solution is the combination of objects that minimises the weight of the rucksack but maximises the value of the items contained within it. Again, a combinatorial problem that is relatively simple for a human to solve is often highly complex for a computer to solve.

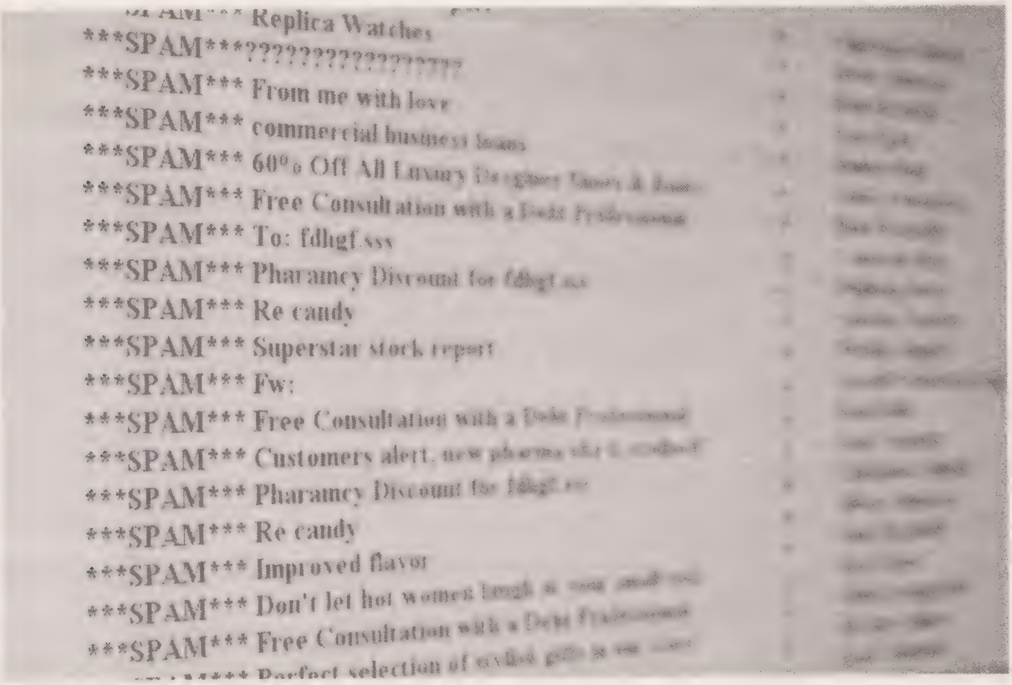


This is one of the many computer systems used for form recognition, in this case face recognition. This image is of a system developed by the Japanese firm NEC.

Learning

The next branch of artificial intelligence is learning. Is a system that can learn from past experience intelligent? Think back to the example of the automated medical diagnosis system, which contains a file with a list of symptoms associated with an illness. This process of adding linked data is called training. So, once the system has been trained, when a new symptom is submitted the system is capable of searching its memory to see whether it has appeared in the past and, if so, responding with the illness that was associated with it. In this case, the system is said to learn through memorisation and is therefore not intelligent. The learning branch of AI is based on training systems so that they are then able to generalise or deduce rules that can then be applied to new problems that have not appeared before.

Machine learning has been one of the most prolific areas of artificial intelligence research. Many universities, research centres and companies are making daily breakthroughs in this field. This is probably due, on the one hand, to the great need for expert systems in industry and, on the other, to the complexity of programming useful expert systems. An intelligent expert system is trained with a series of cases linked to solutions so that it can deduce the rules and norms that link them. When given a new case, the system can then determine the new solution. So, for an expert system to be regarded as intelligent, it is essential that it can learn and generalise automatically, i.e. that rules do not have to be added manually and, once trained, it can behave like an expert in that subject. Later we will look in detail at the applications of expert systems, although we will end now by noting a few illustrative examples of current expert systems, such as those that predict late mortgage payments, detect tumours early and automatically classify unwanted email (or 'spam').



```
Replica Watches
***SPAM***?????????????????
***SPAM*** From me with love
***SPAM*** commercial business loans
***SPAM*** 60% Off All Luxury
***SPAM*** Free Consultation with a
***SPAM*** To: fdhgf.sss
***SPAM*** Pharamcy Discount for
***SPAM*** Re candy
***SPAM*** Superstar stock report
***SPAM*** Fw:
***SPAM*** Free Consultation with a
***SPAM*** Customers alert,
***SPAM*** Pharamcy Discount for
***SPAM*** Re candy
***SPAM*** Improved flavor
***SPAM*** Don't let hot women
***SPAM*** Free Consultation
Perfect selection of
```

The automated classification of email into 'spam' and legitimate messages is one of the applications of expert systems.

Planning

The third main branch of artificial intelligence is planning. This capability has been a human trait since time immemorial; in fact, it is this ability that has ensured our survival throughout history. Early humans in the Palaeolithic era would have come across a dilemma that required planning.

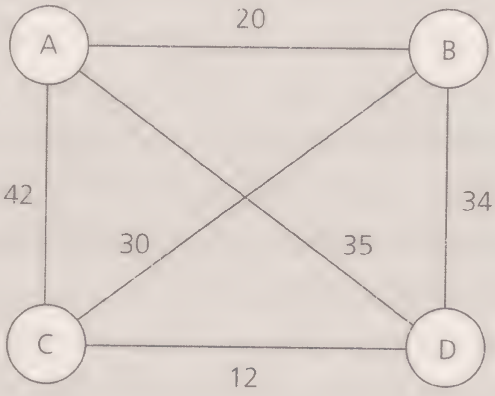
Given a certain amount of food and a certain number of consumers (the members of the group), how should the food be distributed between all individuals? Should they give the most succulent and energy-rich meat to the people responsible for gathering berries or to the strong hunters? And what happens if one of the gatherers is in the late stages of pregnancy? All these questions belong to what we call 'systemic constraints' – circumstances that have to be taken into account when drawing up a plan.

Constraints belong to two basic categories: violable and inviolable. In our previous example of the prehistoric tribe, although the best food should go to whoever needs it most, it wouldn't matter if one day the strongest hunter in the tribe didn't get the most nutritious morsel from the day's hunt. While this situation is unsustainable in the long term, the individual can manage his hunger for one or two days. Therefore, this is a violable constraint.

THE TRAVELLING SALESMAN PROBLEM

A given problem can often be classified as belonging to one or other branches of artificial intelligence depending on what approach is taken to solving it. A good example is the famous TSP (*Travelling Salesman Problem*), which can be solved using a search strategy or a planning strategy.

It is formulated as follows: given a list of cities, the routes between them and their distances, find the shortest possible route that a travelling salesman would have to take to visit customers in each city. The salesman should visit each city exactly once, and cover the shortest possible distance. As readers may have noticed, depending on the layout of the routes between cities, it might be absolutely necessary to repeat a visit to a city in order to visit them all, so repeating a city can be regarded as a permissible violation.



An example of a graph of interconnected cities. The numbers on the edges indicate the distances in kilometres between the cities.

The opposite case may be, for example, the distribution of resources in a large university (classrooms and teachers) when planning an academic year. In this case, the consumers of resources will be the cohort of students registered to take, say, numerical calculus, commercial law, physics, etc. When deciding how to allocate resources, it would have to be taken into account that the commercial law and physics groups cannot share room 455 at the same time. Nor can the professor of numerical calculus give a class on commercial law at any time of the year given that he, in all probability, is not qualified to do so. In these examples, the constraints are said to be inviolable.

Whether constraints are violable or inviolable is critical and is a fundamental aspect to be taken into account when programming an intelligent planning algorithm.

Automated reasoning

The fourth branch of artificial intelligence is automated reasoning. This is, without doubt, the branch that sparks the most interest and fascination in the imaginations of the general public, and is a regular subject of science fiction books and films. However, this field was born out of the decidedly unglamorous automated demonstration of mathematical theorems.

New theorems are frequently being formulated and mathematicians have to prove whether or not they are true, a process that can be highly complex. This is what happened with Fermat's last theorem (which states that if n is an integer greater than two, then there are no non-negative natural numbers that satisfy the equation $z^n = x^n + y^n$). We had to wait more than 200 years for a proof of this theorem!

In 1956, economist Herbert Simon (1916-2001) and engineer Alan Newell (1927-1992) jointly developed Logic Theorist, a machine capable of proving complex mathematical logic theorems. This marked an important milestone in the discipline of artificial intelligence and revived philosophical discussions about the possibility of building thinking machines. There is no doubt that many of the books and films released in the 1960s and 70s that featured malevolent intelligent machines were influenced by these discussions. According to the influential philosopher Pamela McCorduck, Logic Theorist is the proof that a machine can perform tasks regarded as intelligent, creative and once thought only executable by a human.

Herbert Simon (left) and Allen Newell playing chess in 1958.

Logic Theorist used what are known as 'symbolic systems', invented by mathematicians to give meaning to certain expressions without referring to arbitrary conventions. For example, we might say that 'being a man' implies 'being mortal', a sentence that can be formulated using the mathematical expression 'A → B', in which the symbol 'A' is equivalent to 'being a man', the symbol '→' signifies 'implies' and 'B' is equivalent to 'being mortal'. 'Being a man implies being mortal' is an arbitrary expression that is formalised using the expression 'A → B'. Once the arbitrary terms are formalised, it is much easier to manipulate them and carry out operations with them from a computing or mathematical perspective.

With the aim of simplifying mathematical operations, symbolic systems use axioms as a departure point for constructing theorems based on rules of inference. The advantage of symbolic systems is that, being clearly defined, unambiguous formal systems, they are relatively simple to programme. Take an example:

Socrates is a man.
All men are mortal.
Therefore, because Socrates is a man, he is mortal.

Formulated mathematically, these sentences would take the following form:

A: Socrates

B: man/men

$A \rightarrow B$

C: mortal

$B \rightarrow C$

If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$, i.e. Socrates is mortal.

In this example, a rule of inference known as a 'hypothetical syllogism' leads us to conclude that $A \rightarrow C$ if it is true that $A \rightarrow B$ and $B \rightarrow C$.

However, the automated and systematic inference of theorems from axioms and rules can lead us to a number of combinations that, once again, comes dangerously close to the number of atoms in the Universe. For this reason, Logic Theorist used heuristic considerations. Put another way, these are loosely predictive instruments that help to select the best of all the possible inferences, in order to identify the correct sequence of inferences that need to be made on the basis of the axioms to prove the theorems.

Take a practical example. We want to know whether Socrates is mortal or not, and we know the following initial axioms:

A: Socrates

B: supporter of Olympiakos

C: Greek

D: man

E: mortal

$A \rightarrow C$

$C \rightarrow D$

$A \rightarrow D$

$C \rightarrow B$

$D \rightarrow E$

And we want to find out whether A → E is true or false using brute 'force', by proving all the possible combinations; this gives us:

A → C → D → E
A → C → B
A → D → E

We have thus performed seven logic operations based on just five axioms, using only one rule of inference: the hypothetical syllogism. As you can imagine, in more complex scenarios, with more axioms and more rules of inference in play, the number of possible combinations can be so high that it would take years to produce conclusive proofs. Had we used Simon and Newell's methodology to solve this problem, a heuristic consideration (or 'a heuristic', as the experts abbreviate it) would have told us we were on the wrong track if, in order to demonstrate that someone is mortal, we had to begin by talking about football (A → C → B).

Symbolic systems and heuristics are now widely used to solve practical problems, not just in automated mathematical theorem proof systems.

To illustrate another use of heuristics, let's go back to chess. For every turn in chess, there are on average 37 possible moves. So, if a computer program tried to analyse a play eight moves into the future, it would have to analyse the equivalent of 37^8 possible scenarios, in other words 3,512,479,453,921 moves – that's more than 3.5 billion. If the computer spent one microsecond analysing each play, this would mean that just to analyse eight moves into the future (something an expert player would regard as quite simple), a powerful computer would have to think for more than two and a half years before each turn!

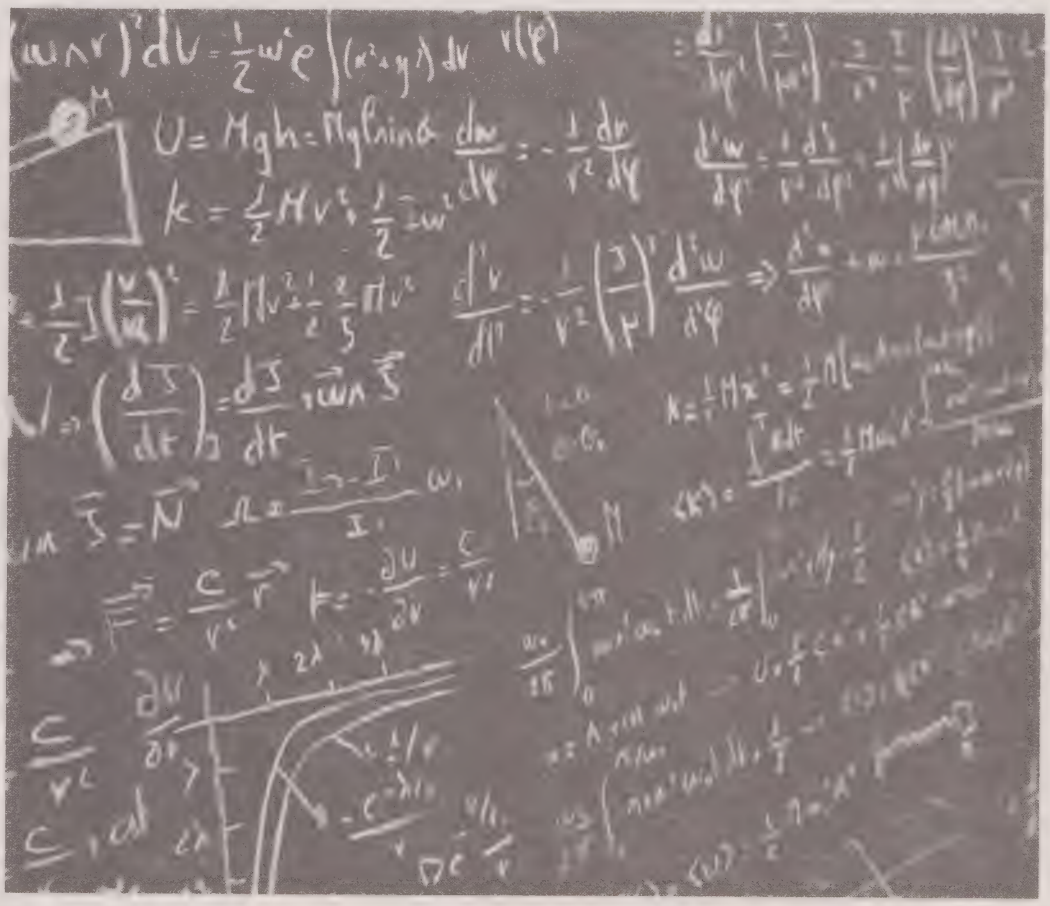
This suggests that we need to make some type of improvement to the method to speed up the process, and this improvement involves heuristics. These are predictive rules that help the algorithm to rule out those plays that, for some reason, are perceived to lead to very unfavourable situations, so there's no point exploring them further. Just by eliminating from the analysis a few absurd plays, the saving in terms of moves requiring analysis can be enormous. In summary, heuristics are predictive tools based firmly on the programmer's intuition and are so fundamental to most intelligent systems that system quality largely depends on them.

In recent years, gradual progress has been made in automated reasoning and reasoning is now possible in incomplete, uncertain and 'grey-area' or non-

MATHEMATICAL LOGIC

Mathematical logic is a branch of mathematics that involves studying forms of reasoning. In other words, it is a discipline which, using rules and techniques, determines whether or not an argument is valid. Logic is widely used in philosophy, mathematics and, of course, computing as an instrument for validating or generating new knowledge. George Boole, with his 'Boolean' algebra, and Augustus De Morgan, with his logical laws, used the basics of Aristotelian logic and a new more abstract notation to develop a useful instrument for exploring the fundamentals of mathematics.

In the last 50 years mathematical logic has made great progress and given rise to what is termed 'modern logic'. To distinguish it from classical logic, the latter is referred to as 'first-order logic'. Formally, first-order logic only involves finite expressions and clearly-defined formulae. There is no room for infinite fields or uncertainty.

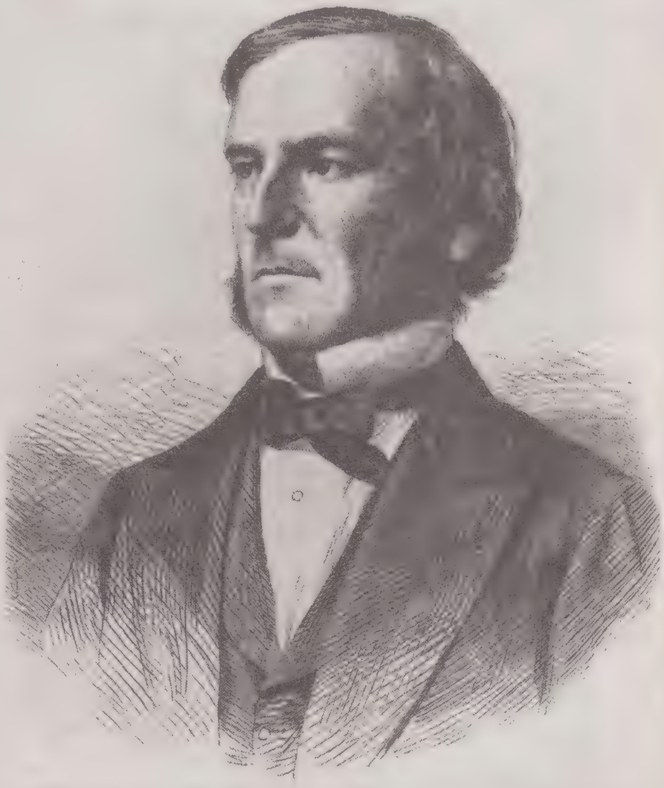


However complicated an expression written on the blackboard may seem, it is very rare that a symbol will be used that is outside the scope of first-order logic.

monotone systems, i.e. environments that lack information (incomplete) can make initial contradictory statements (uncertainty) or, when bringing new knowledge in, overall knowledge of the environment does not necessarily increase (non-monotony).

One hugely powerful tool that can work in these environments is fuzzy logic, which is a type of mathematical logic in which statements don't have to be completely true or false. Whereas in 'classic' Boolean logic a statement can always be said to be true or false (for example, it is false that 'some men are immortal' and it is true that 'all men are mortal'), in fuzzy logic there are grey areas between truth and falsehood. So, previously, if we'd said that Croesus wasn't poor, it would have automatically followed that he was rich, and if we'd said that Diogenes wasn't rich, it would have

GEORGE BOOLE (1815–1864) AND HIS LOGIC



If Alan Turing can be said to be one of the fathers of modern computing, George Boole could be said to be one of its grandfathers. Indeed, this British mathematician and philosopher developed Boolean algebra, the basis of modern computer arithmetic, on which in turn the whole of digital electronics is based.

Boole devised a system of rules that, using mathematical procedures, enable the expression, manipulation and simplification of logic problems that permit two states: true or false. The three basic mathematical operations in Boolean algebra are negation or complement ('NOT'), disjunction ('OR') and conjunction ('AND'). Negation, represented using the symbol ¬, consists in reversing the value of a variable. For example, if A = "Aristotle is a man", then ¬A = "Aristotle is not a man". Disjunction, represented using the meant that he was poor (in this example, Boolean logic discriminates against the comfortably off!). But now, using fuzzy logic, we can say that Aristotle is rich to a degree of, say, 0.6.

Natural language processing

Understanding a human's natural language, and being able to communicate using it, is one of the great remaining challenges in this area of science. If an artificial system could understand a human using his or her natural language, whatever language that may be, without requiring programming languages, it would go a long way towards humanising machines, regardless of their

symbol $\vee$, is a binary operator, i.e. it needs two parameters to give the result. It is true if either of the two parameters is true. For example: "Is it true that what you're doing now is reading OR driving?". The answer is "Yes, it is true", because what you're doing now is reading this book. If you were currently driving and not reading, then the answer to the question would still be "Yes". It would equally be true if you were taking the extremely unwise step of driving and reading this book at the same time. Finally, the third operator is conjunction, represented by the symbol $\wedge$, which is also a binary operator. If we were to reformulate the last question and ask "Is it true that what you're doing now is reading AND driving?" we could only answer "Yes, it is true" if we were doing such a foolish thing.

Based on these three operators, we can build other, more sophisticated operators, such as exclusive or ($\overline{\wedge}$), which would only be true, in the case of the second question, if we were reading or driving, but not both at the same time. The operator $\overline{\wedge}$ is not a basic Boolean operator because it can be rewritten using the three basic operators: $A \overline{\wedge} B = (A \wedge \neg B) \vee (\neg A \wedge B)$.

In digital electronics, 'true' and 'false' are symbolised by 1 and 0 respectively, and signify whether the electrical current is allowed to pass (1) or not (0). Logic operations are performed by combining transistors and it is the deep integration of these electronic systems that has enabled computing to make such progress in the last 40 years. The microprocessor in your home computer, which is the device's core brain, contains hundreds of millions of transistors laid out in a space measuring a few square centimetres!

intelligence. However, the ability to "process natural language" does not just mean transliterating sounds into phonemes and written symbols, but also understanding the concepts being conveyed (in written or oral form) in order to perform some action on that basis.

The first progress in the processing of natural language was made by engineer Joseph Weizenbaum (1923-2008) between 1964 and 1966 using his Eliza program. Users can ask Eliza questions and make comments (she's a little like a psychotherapist), and she seems to understand them, because she's able to respond in a more or less meaningful way. A real conversation the author had with Eliza while writing this book went like this:

Me: I need a holiday.
Eliza: What does having a holiday mean to you?
Me: Well I could spend more time with my wife.
Eliza: Tell me more about your family.
Me: Well, I only have one wife.
Eliza: You say you only have one wife?
Me: Yes, I'm not a polygamist.
Eliza: You seem very positive.
Me: Her name is Laia.
Eliza: Who else is there in your family?
Me: Nobody else.
Eliza: Really, nobody else?
Me: No.
Eliza: Are you sure?
Me: Very sure.

Psychologists can breathe a sigh of relief: Eliza will not be putting them out of a job any time soon. However, since the 1960s natural language processing has made significant progress and there are now natural language processing techniques that are used for applications as diverse as detecting literary plagiarism, automatically translating texts and constructing more humanised person-machine interfaces. If you're curious, you can talk to Eliza yourself at:

http://www.chayden.net/eliza/Eliza.html.

Managing what we know

Structuring knowledge properly is very important, and to illustrate why let's consider an example. Imagine someone asks you to name the mayor of Austin, Texas. If you're not from that part of the United States, you'd probably answer: "I don't know". Conversely, a system that does not have its knowledge well structured, as may be the case with our personal computers, will spend a few minutes analysing all the documents on its hard drive in search of a document that contains the name of the mayor of that city. An intelligent system, or one that aspires to be intelligent, should answer as categorically as a human being would and, for that reason, knowledge should be well structured and easy to access.

When solving a practical problem, not only should knowledge be properly structured, the right tools also need to be programmed to navigate through it and keep it ordered. This knowledge base is where the system applies its reasoning, research, learning and other strategies, so the knowledge base of an intelligent system is subject to change.

Intelligent systems therefore need knowledge control engines which resolve, for example, the contradictions that can keep cropping up, eliminate redundancies and also generalise concepts.

To keep proper control over the knowledge contained in a knowledge base, meta-information is needed to explain how the knowledge that helps us to delimit it is represented internally. Knowing how knowledge is represented is no easy task, as it can be structured in a wide variety of forms. So, having information about how the stored knowledge is structured can really help automated systems to navigate through it.

Another aspect that has to be taken into account is how knowledge is delimited, because knowing what the knowledge base covers and its limits also helps the computerised system to navigate it. A human being can easily cope with the idea that his or her knowledge is incomplete, but a computerised system needs to be told very clearly what it does and doesn't know. One of the first methods devised for managing knowledge bases was the Closed World Assumption (CWA). The CWA was proposed by Raymond Reiter in 1978 and is based on a simple statement but one with significant consequences: "The only objects that can satisfy the predicate P are those that must necessarily do so" or, in other words, any knowledge that is not stored is not true.

Imagine you were asked whether a certain person works for a company. To find out you could look at that company's staff list, and if the person was not on it you would say that she doesn't work there.

The CWA was a major breakthrough and made knowledge base management much easier. However, as you will have noticed, the CWA has major limitations, since in real life not knowing something does not automatically mean it is not true. Going back to the example of the company staff list, what happens if a person who works for a company is not on the list of employees for the simple reason that the list contains an error or has not been updated since their appointment? This is in fact one of the main weaknesses of the CWA – the inaccuracy or inconsistency of real-life data. Its other weakness is that it obliges us to use purely synthetic reasoning. This is illustrated in a practical example:

Imagine you have the following list of single and non-single people:

John is single
Mary is single
David is non-single

If someone asks the system whether George is single, as this is a list of single people, it would answer no, as he is not on the list. However, you could make up a new list of married people:

John is unmarried
Mary is unmarried
David is married

And if the system were now asked whether George is married, it would also answer no. Which in the end goes to show that, because it has no information on George's marital status, the system comes to the incongruous conclusion that he is neither married nor single. Clearly, the CWA does not function well in cases where knowledge is incomplete or uncertain and, therefore, nowadays it is only used to solve very particular problems.

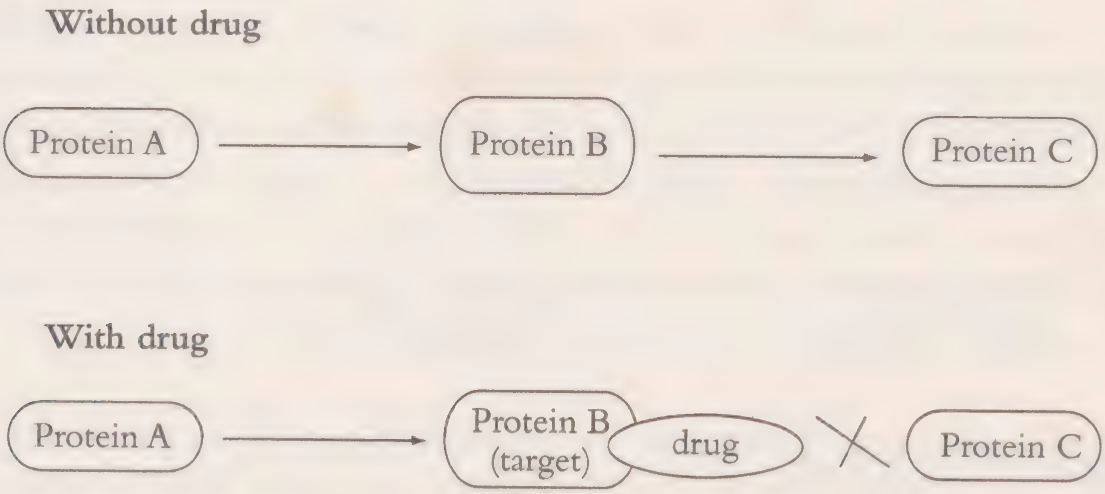
Finally, we couldn't finish our section on knowledge management without talking about Truth Maintenance Systems (TMS). TMS are systems that monitor and check that the knowledge base is internally consistent, and they are especially useful when non-monotone reasoning methods are used, i.e. methods where the knowledge base increases and decreases in size as it reasons. TMS come in two different types: 'vertical search' and 'horizontal search'. The former scour the knowledge base from the general to the particular in search of contradictions and, if they find any, retrace their steps in search of a solution. Conversely, horizontal search systems posit various parallel scenarios or hypotheses, so that the Universe of contexts is gradually pruned as contradictions are discovered. This means that, given a possible context (think of a specific chess position), it identifies the different scenarios in which the current situation might change (in the case of chess, the possible moves) and eliminates those which are contradictory (in chess a contradictory scenario would be a play with very unfavourable consequences for the machine, given that the aim is to win the match and it would be a contradiction to consider a play that is unfavourable for the machine).

Chapter 2

Research

How is a new medicine designed? Until very recently, pharmaceutical companies did it completely by hand, in other words, without any technological assistance other than pencil and paper. They designed and optimised the chemical structure of the drug and synthesised and tested each improved version in specialist laboratories to check whether or not its effectiveness increased. This fully manual process of trial and error is partially responsible for the fact that it costs approximately $1 billion to develop a new drug.

When a new drug is designed, what is generally being developed is a molecule that can interact with a protein and, potentially, inhibit its action. The proteins act in living things, setting in motion what are known as 'metabolic cascades' – series of biochemical reactions mediated by the same proteins. Therefore, if a molecule inhibits the expression of one of the proteins involved in the metabolic pathway in question, this molecule will block the pathway and might be a good drug.



In this simplified scenario, the drug impedes the interaction of the target protein with protein C, thereby blocking the metabolic pathway.

To successfully inhibit one of these proteins, the drug must be made to bond with one of them in a certain way. For this reason, much of the drug design and development process focuses on bonding the molecule with the active centre of the protein in question, also known as the therapeutic target.

To discover whether a molecule bonds effectively with a protein, the interaction energy has to be measured. The interaction energy between a molecule with potential to be made into a drug and its target protein is equivalent to the energy that has to be applied to the system to maintain the bond. For example, if you wanted to bond a fridge magnet to the door of the fridge, you wouldn't need to apply any continuous force, because the magnet bonds with the metal of the door by virtue of its magnetic properties. When the magnet is really powerful, you only have to move the magnet close to the door to see how it is attracted to the door with a degree of force. In this case, the energy that has to be applied to keep the magnet and the door united is said to be negative. Both objects are attracted to each other.

As you can imagine, a drug that is not attracted to its target protein has no value in its current state, because it will merely float through the bloodstream or the tissues, ignoring and being ignored by the protein it should be inhibiting. So, when looking for a good potential drug, researchers try to find a composite whose interaction energy is as negative as possible, because this is a sign of how strong the attraction between drug and protein will be. Therefore, the main optimisation aim in the process of designing a new drug is to minimise this bonding energy.

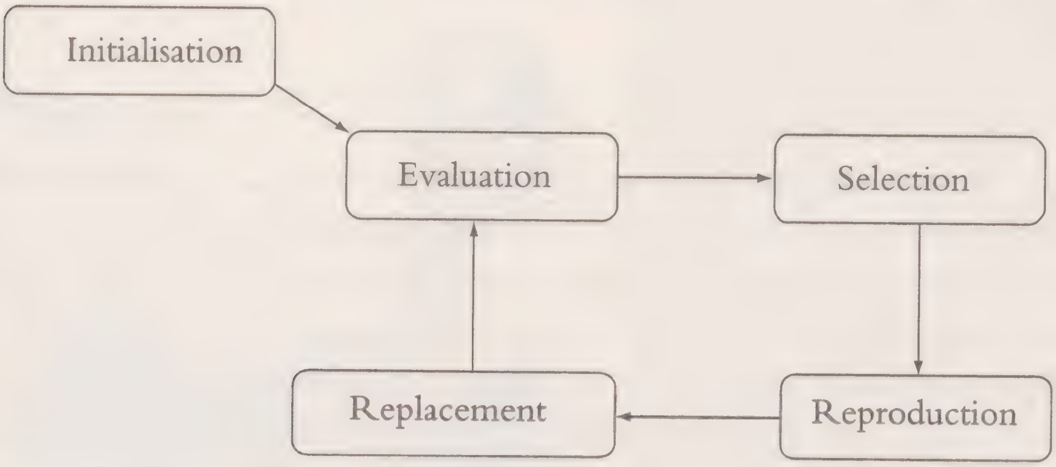
This type of problem, where the solution consists in determining optimal parameters (whether identifying which chess piece should be moved at each turn to win the game, or the dimensions and positions of each girder in a bridge in order to minimise cost and maximise resistance), belongs to the category of 'search problems'. Search is one of the main branches of artificial intelligence. In a search exercise, the aim is often to find the parameters that maximise a mathematical function and, in this particular case, search is also known as 'optimisation'.

Darwin said it first

One of the techniques used most often to solve search problems is evolutionary computation. In the same way as nature has evolved her living things in order to maximise their chances of survival in their respective natural environments, evolutionary computation uses similar mechanisms to optimise functions of varying complexity.

Evolutionary computation was first proposed by researcher John Holland in 1975 in his book *Adaptation in Natural and Artificial Systems*, although Western science later discovered that German engineers had already used these strategies to optimise the nozzles in early jet engines during the Second World War. Evolutionary computation covers a wide range of evolutionary techniques or algorithms inspired wholly by Charles Darwin's laws of natural selection, according to which the best prepared individuals are those that exhibit the best survival rates and, therefore, have the most viable descendants.

In this metaphor inspired by the laws of evolution we have populations each of whose members represents a possible solution to a problem. And, whether it's a good or a bad solution, what evolutionary algorithms try to do after evaluating the fitness of each member is to select the best ones and use them to produce a second generation. In an iterative process, the members of successive generations are evaluated, selected and crossed to produce new populations or generations at each turn. The end of this process is determined by stop criteria that may vary depending on the problem. So an evolutionary algorithm comprises five main stages: initialisation, evaluation, selection, reproduction and replacement, as shown below:



The differences between evolutionary and other algorithms are determined by the various ways in which each of these main stages is implemented.

Initialisation

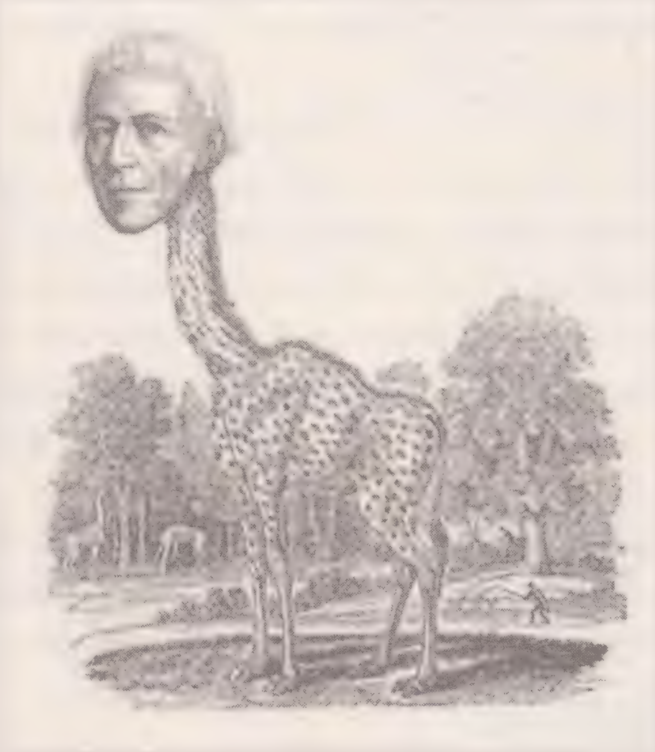
The population initialisation stage is not really governed by the evolutionary algorithm used. In fact, it depends more on the characteristics of the problem in question.

DARWIN AND LAMARCK: TWO DIFFERENT VISIONS OF EVOLUTION

Jean-Baptiste-Pierre-Antoine de Monet, Chevalier de Lamarck (1744–1829), was a French naturalist who revolutionised biology. He made important contributions including the classification of living things according to their complexity and the definition of a clear dividing line between the organic and inorganic worlds. Another of his contributions to science was the first theory of biological evolution, described in his book *Philosophie Zoologique (Zoological Philosophy)* in 1809, 50 years before Darwin's theory of evolution was published.

Lamarck's theory, unlike Darwin's, is based on the 'inheritance of acquired characteristics', in other words individuals' ability to pass on to their descendants the environmental adaptations they have acquired in life. A good example which clearly shows both perspectives on evolution is the giraffe's long neck. According to Lamarck, the giraffes that stretched their necks the furthest and developed their muscles the most in order to reach the highest leaves passed this characteristic on to their descendants, which in turn continued to develop these muscles and pass them on to their descendants until the giraffe's neck reached its current length. According to Darwinism, on the other hand, giraffes born with longer necks or more powerful neck muscles managed to pass these characteristics on to their descendants, regardless of the efforts they made in life.

Although Lamarck's hypotheses were rejected as false in favour of Darwin's, they have recently been acknowledged to have some veracity in a number of specific cases. For example, we know that a mother who has developed antibodies in the process of getting over an illness can pass these antibodies on to her descendants, so that her offspring will also be immune to the illness. This is a case of the transmission of characteristics acquired in life as an adaptation to the environment.



Caricature of Lamarck as a giraffe.

Some problems impose constraints to be taken into account. In others, absolutely nothing is known about what a good solution should look like and, therefore, the problem is initialised in a completely random way; and there are others where it is preferable that initialisation is random, but postulate that the individuals generated in this first generation will have a certain guaranteed level of diversity, so that no stone is left unturned.

The decision on how knowledge within an individual should be represented is especially important at this stage, because it will largely determine the remainder of the evolutionary algorithm. One of the most common representations uses chromosomes, a new concept inspired by nature: a chromosome is a sequence of genes, and each gene is a number that represents part of a solution.

Take the example of an algorithm that seeks to maximise the capacity of a cardboard box while minimising the quantity of cardboard used to manufacture it. If an evolutionary algorithm is used, the chromosomes that would represent the solution would have three genes: length, width and height. Therefore, at the initialisation stage, a population of random boxes is created, represented by lists of three numbers within the permitted ranges, and the algorithm evolves the box populations until it finds the optimum box according to the established criteria.

Evaluation

After initialisation comes the evaluation stage, which is usually said to be the most important stage in the process as it defines the problem to be solved. The first step in the evaluation stage is to reconstitute the solution. For each individual, the information from its chromosome (genotype) is taken to simulate the represented solution (phenotype). This process may comprise different levels of complexity, from simply calculating the volume of a box on the basis of its dimensions, as with the cardboard box problem, to extremely costly and complex calculations, such as simulating the resistance of a bridge during its design process.

Once the phenotype is reconstructed, the fitness of this solution is evaluated and a fitness value is assigned to each individual; in subsequent evolutionary stages this value will be used to distinguish between good and bad solutions. Once again, the phenotype evaluation process itself may be complex, costly and even noisy; by this we mean that when solving a number of complex problems, a single phenotype evaluated on several occasions may not always produce the same fitness value.

Noise, which we might also call 'error', is a constant factor in problems where fitness evaluation requires numerical simulations. For example, when simulating the resistance to wear of a combustion engine part, solving mathematical equations that determine this wear would be so expensive that the best option is to use a simulation process, which would probably deliver somewhat different results in each separate simulation of one part. (In 2004 Honda presented the results of tests that used genetic algorithms to design engine parts. The evaluation process was not only noisy and a little imprecise, but also slow. It took eight hours to calculate the fitness value for each individual in the population.)

Selection

The next stage in an evolutionary algorithm, after the evaluation of individuals in the current generation, is selection. The idea behind selection is to choose the best individuals that will reproduce and generate descendants for the next generation. This process of selecting the best individuals is also the basis for evolution in nature and is referred to as 'evolutionary pressure'. Evolutionary pressure increases as

THE PLUMP MAURITIAN BIRD AND EVOLUTIONARY PRESSURE

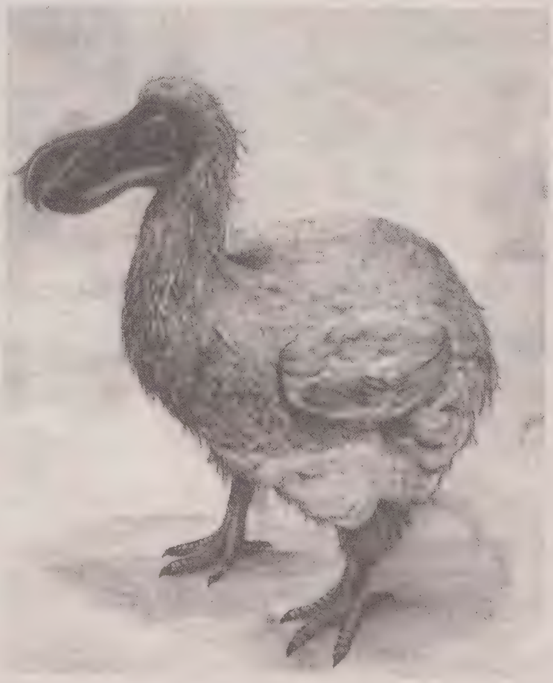
When explorers first arrived in Mauritius in the 17th century, they discovered a surprise gift from the heavens – a plump bird with succulent flesh, wings too small for it to fly and legs too short for it to run away. They called it a "dodo". Settlers ruthlessly hunted it, and the explorers' pets (cats and dogs), along with other new species brought to the islands, like rats, destroyed their nests to eat their eggs. The hapless dodo was extinct in a little less than a century and only a few drawings and engravings of this gentle and inoffensive bird remain.

The dodo had never 'needed' to evolve, had never been subject to evolutionary pressure, and when it was subjected to that pressure it didn't have time to cope with it. Evolutionary pressure is the driver of evolution. Without a degree of evolutionary pressure, living things do not have enough reason to adapt to the environment, and don't need to develop an optimal form, behaviour or appearance. The history of the natural sciences is littered with examples of species that were clearly in this situation. Species immersed in an environment with plenty of food, a lack of predators or little competition between species, making it difficult for them to develop characteristics that the proportion of individuals that go on to the next generation reduces. However, if a strategy as simple as directly picking the best individuals is applied, it can be demonstrated that the evolutionary pressure applied is too high and evolutionary algorithms don't usually cope well with excessive evolutionary pressures, often falling into local maxima.

The main benefit of an evolutionary algorithm is that it can find good solutions in large search areas or, in mathematical terms, optimise functions that are normally multidimensional and 'multimodal', i.e. they have varied local or global maxima. If excessive evolutionary pressure is applied, in other words if we try to find the solution too quickly by selecting the best individuals immediately without looking any further, the algorithm will converge prematurely and fall into local maxima.

Selection is the ideal stage for modulating the evolutionary pressure of an evolutionary algorithm. The pressure will be most extreme if a unit selection is made, i.e. if only the best individual in a population is selected as a basis for generating the individuals in the next generation. At the other extreme is random selection, which does not take account of individuals' fitness at all. Logically, the appropriate strategy to follow will fall halfway between the two, where we try to select the best

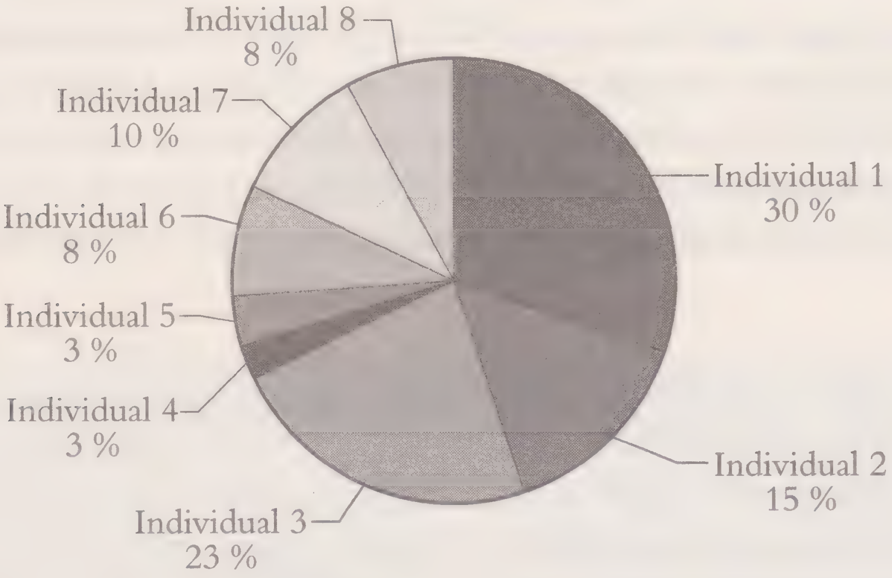
are essential for other similar species in more competitive environments. This could be said to have happened to the dodo. With no predators and no shortage of food in its protected island ecosystem, it had no need to retain working wings or long legs so it could run quickly. In fact, the literal translation from Portuguese of the word *dodo* is "stupid". Could it be that the absence of evolutionary pressure made this animal 'stupid'?



A dodo in a 17th-century engraving.

individuals so that they can reproduce, but still introduce a certain degree of diversity in order to explore other avenues. This strategy ensures that there will always be some probability that an individual, however unfit it is, will be selected even though there are much better individuals in the population. The three selection strategies that behave in this way, and are also currently the most commonly used strategies, are known as roulette, ranking and tournament selection.

Roulette wheel selection is a fairly simple system in which each individual has a selection probability proportionate to its fitness value in relation to the fitness of the other individuals. Therefore, if ten individuals have to be selected, the roulette wheel is spun ten times.



In the example in the diagram there are eight individuals and each has a fitness proportionate to the total fitness of the selection, indicated in each wedge. Each time we spin the roulette wheel, the probability that a given individual will be selected is proportionate to its fitness in relation to the total, but the roulette wheel strategy does not rule out the selection of less fit individuals; it is just less probable that this will happen. If we spin the roulette wheel ten times, it is certain that fit individuals will be selected on some occasions, but it is also probable that on some occasions a less fit individual will be selected. This probability that uncompetitive individuals will be selected is what gives genetic algorithms so much power, because it allows them to follow different avenues at the same time, and explore other areas. As a result they can identify a large number of different maxima and determine, in the long run, a good local maximum or, in the best case, the global maximum.

Another selection system with the ability to solve complex problems is ranking selection. It is a fairly simple system: it involves selecting n copies of the best individual, $n-1$ copies of the second-ranked individual, and so on as far as $n=0$. This system eliminates the possibility that a 'superindividual' might eclipse the selection possibilities of any other individual. An individual is known as a superindividual when, although far from being an optimum, it is far fitter than its generational companions. Therefore, the population stagnates around it and the algorithm is not capable of improving its qualities.

However, the third method, tournament selection, has achieved a monopoly on selection policies used to solve real problems because it has sound mathematical properties and offers very versatile options for modulating evolutionary pressure. Tournament selection functions in the same way as pairings in a sporting competition. Random pairings of individuals are selected, two-by-two, and the winner of the tournament is regarded as the best individual and is selected. Therefore, as many pairings need to be put together as there are individuals to be selected. But why do we say that the tournament method is so versatile in terms of modulating evolutionary pressure? Well, what would happen if instead of organising tournaments between two individuals we played off n individuals? And if instead of having only one winner per tournament there were m winners? In this case it is said that tournaments are in an n/m set-up, and the greater the value of n, the more evolutionary pressure is exerted, and the greater the value of m, the lower the evolutionary pressure.

For a better idea of the tournament structure, think of the initial group stages of the UEFA Champions League. In this case, the tournaments are in a 4:2 set-up, i.e. four football teams are drawn at random and only the top two are selected to go through to the next phase of the competition. In reality, the Champions League is not a strictly random tournament, because certain selection rules apply to the four initial teams. For example, two teams from the same country cannot meet. Similarly, we can also apply our rules to evolutionary algorithms, which will result in one type of evolution or another.

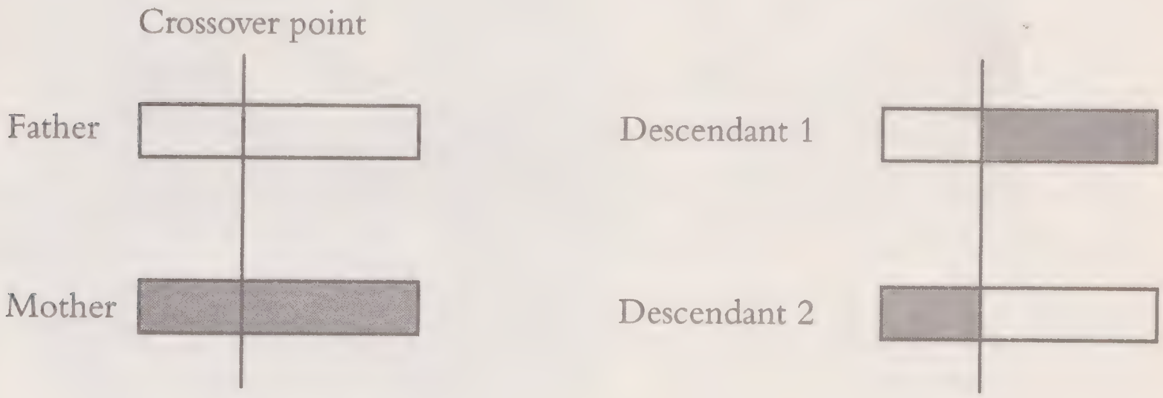
A quite commonly used rule is that only the most similar individuals can compete in the same tournament. The algorithm is therefore capable of optimising functions with many optima.

These 'robot crabs' are light seekers. One has no legs, while the other has four. Both were used by Josh Bongard of the University of Vermont, who equipped them with an evolutionary algorithm and could check that his machines, once evolved, functioned better than normal robots devised explicitly for the same purpose.

Reproduction

After the individuals that are going to have descendants have been selected, the next stage is reproduction. There are various types of reproduction systems and, although it is not necessarily the most important part of an evolutionary algorithm, in reality each algorithm is defined by its reproduction system. In other words, a specific evolutionary algorithm is given its name according to the type of reproduction that it uses. For example, genetic algorithms, which we will discuss next, are evolutionary algorithms that use a crossover reproduction system with mutation.

Genetic algorithms are the most used evolutionary algorithms because of the good balance they strike between programming difficulty and meaningful results. Reproduction through crossover and mutation is strongly based on the concepts of actual genetics. In a genetic algorithm, each individual solution is represented by a chromosome, and each chromosome is a sequence of genes. When the parents' chromosomes are crossed, a random crossover point is first created which divides them into two halves. Next, these four halves (two for each parent) are swapped between the parents to generate two descendants. The first descendant contains the first segment of the 'father's' chromosome plus the second segment of the 'mother's' chromosome, and the second descendant is a chromosome formed by the first segment of the 'mother's' chromosome, as far as the crossover point, and the second segment of the 'father's' chromosome.



One final point on genetic algorithms. Once the descendants have been created, they enter a mutation process in which, with a very low probability (typically around 5%), the values of the genes that make up the new chromosomes are changed randomly. Both in practice and in theory it can be demonstrated that, without mutation, genetic algorithms are not good drivers of optimisation because they usually stagnate at local sub-optima or maxima. Mutation enables genetic algorithms to proceed in small random jumps within the search space. If the results of these random jumps are not promising, they will be lost in the evolutionary process, but if they are positive, they will be adopted by the fit individuals in subsequent generations.

Replacement

Finally, the stage that closes the cycle of the evolutionary process is replacement. The aim of this stage is to select which individuals from the previous generation will be replaced by new individuals generated through reproduction. The most common strategy involves replacing all individuals from the previous generation except the

GREGOR MENDEL AND GENETICS

Gregor Mendel (1822–1884) was an Austrian monk who discovered and published, in 1866, the first laws of genetics. These laws, now known as Mendel's laws of inheritance, described the transmission of certain characteristics from parents to their children based on a study which crossed different species of pea. These laws introduced an essential principle to genetics and science in general – the existence of dominant genes and recessive genes.

Mendel began his experiments by making observations of the colouring of the seeds produced by different pea plants. The first generation was obtained by crossing a plant that produced yellow seeds with another that produced green seeds. He then observed that the plants produced by this hybridisation only produced yellow seeds. However, he later observed that although the generation produced by crossing these plants produced mostly yellow seeds, some plants in future generations surprisingly returned to producing green seeds. The ratio between yellow and green seeded plants was 3:1. After performing similar experiments with other characteristics, Mendel concluded that some genes were dominant over others, so that they obscured the existence of the dominated (or recessive) gene and did not allow it to express itself in the individual. This explained why crossing individuals with the same expressed gene could produce descendants in which a different gene was expressed. Without knowing it, both parents possessed this gene, but it was 'obscured' by the dominant gene.

Although the significance of Mendel's work was not recognised in his own lifetime, it provided the basis for genetics, the branch of science that looks at genes and the transmission of characteristics from parents to offspring, and which has been absolutely fundamental to modern biology and medicine.

best, which is given the opportunity to 'live' for another generation. Despite this method, known as 'elitism', being very simple and unnatural, it has been shown to have extraordinary power.

However, many other individual replacement strategies have been proposed. It is worth noting that, again, as in the selection stage, depending on how the individuals to be replaced are selected, the pressure of the evolutionary process can be modulated. If all individuals in the population are always selected and replaced by new individuals, no evolutionary pressure is applied; conversely, if only the poor individuals from the previous population are selected for replacement, the pressure increases enormously.

On the other hand, speciation policies can also be effectively applied at this stage with methods that make it easier to identify various solutions to problems that have various optima. The most commonly used method is replacement via *niching*. This strategy consists in selecting, for each new individual generated, the individuals from the previous generation that most resemble it. In the subsequent generation, only the best individual from the group of similar individuals will be permitted to remain

LAMARCKIAN EVOLUTIONARY ALGORITHMS

The Darwinian–Lamarckian dualism in the evolution of nature also exists in evolutionary algorithms and both methods have in fact proved to be highly effective in solving numerical optimisation problems. Darwinian evolutionary algorithms are the more common (they are the ones we have been describing throughout this chapter), while Lamarckian algorithms include an additional step between evaluation and selection. This step consists of a short local optimisation which simulates an individual's learning or adaptation to the environment before having descendants.

This local optimisation stage is normally based on small mutations that apply to each individual. The individual's fitness is then reappraised to see whether the mutation has made an improvement. If so, it is accepted and the mutation–evaluation cycle is repeated, but if the mutation has reduced the individual's fitness, it is discarded and the mutation–evaluation cycle is repeated, beginning with the state prior to the current mutation.

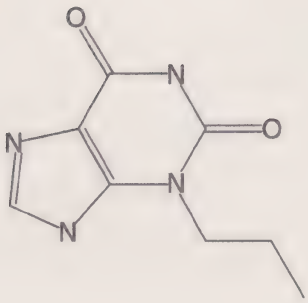
The first Lamarckian evolutionary algorithms were termed 'evolutionary strategies' and, as we have seen, Germany used them during World War II as a tool for optimising the nozzles in their jet aircraft engines.

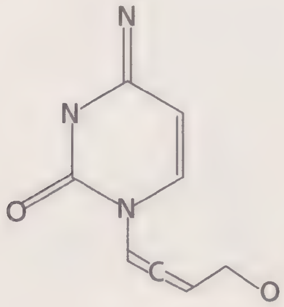
in it. Thus far, we have explained some of the most commonly used methods for completing each of the evolutionary stages. However, it is important to note that there are a host of methods for completing each stage.

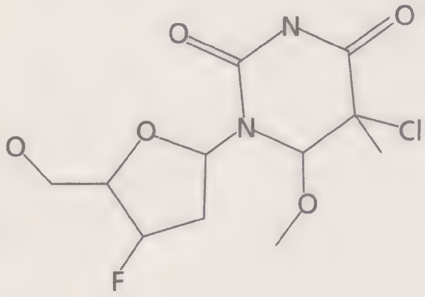
A practical example: evolving a good drug

As we have seen, artificial intelligence offers optimisation methods based on natural processes that deliver great results. Recently, evolutionary computation has been applied to the scientific niche of drug development with notable success. Remember that in drug design the aim is to produce a composite whose energy in bonding with a given protein is as negative as possible, so that an irresistible attraction binds them within the human body like a sweet to a greedy child.

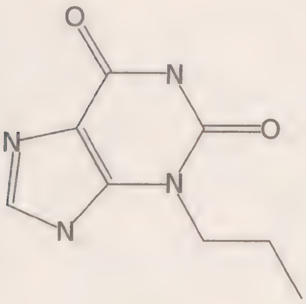
Let's look at how an evolutionary algorithm would act in the process of optimising a candidate drug. Firstly, the algorithm has to initialise the molecule population. At this stage, what usually happens is that a random generation of molecules is proposed. To simplify the example, we will make generations of just three molecules, although normally the generations would contain hundreds:



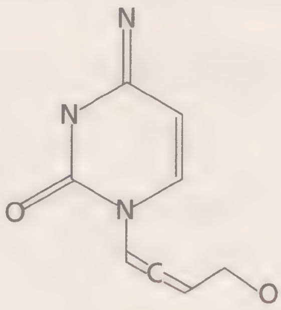




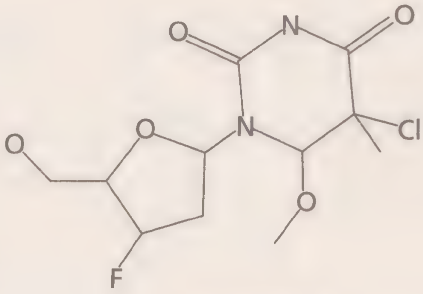
Next, these initial molecules have to be evaluated by estimating the interaction energy between each molecule and the target protein. Various computational tools can be used for this purpose. One of them (which we will only mention, and not explain) is *docking*, a three-dimensional simulation process that predicts how the molecule will behave when it encounters its target: whether or not it will fit with it and what the bonding energy will be. This gives rise to the curious situation where we are using an evolutionary algorithm to find our perfect molecule, but at the same time using an evolutionary algorithm to evaluate the fitness of the molecule with respect to the others. Having completed the *docking* process, we now have our evaluated molecules:



E = 9

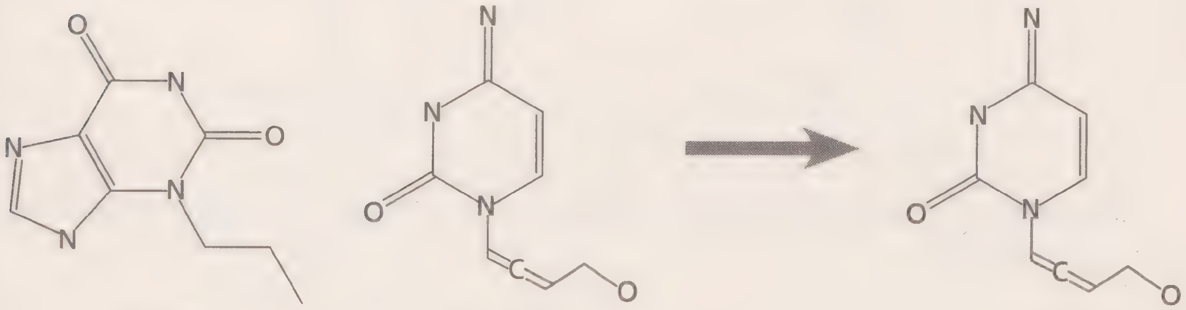


E = –5

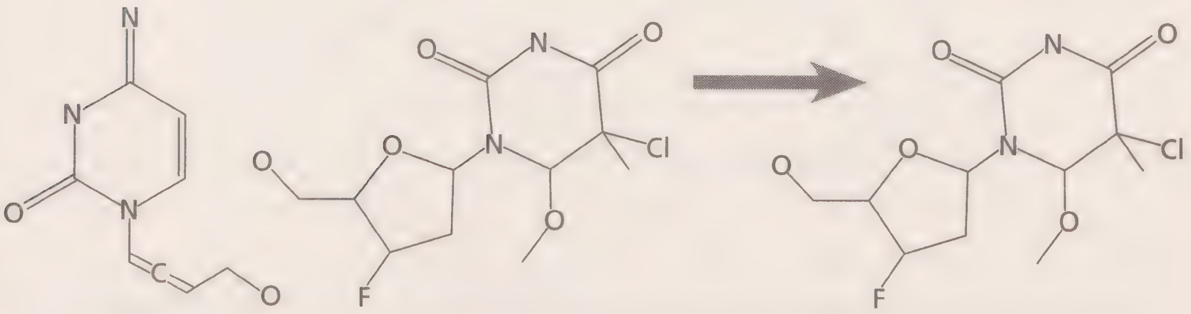


E = –8

The next stage is selection, which we will conduct using a molecular tournament, in which we select pairs of molecules and play them off against each other, comparing their interaction energies and deciding whether they 'stay' or 'go'. Remember that the interaction energies need to be as negative as possible.

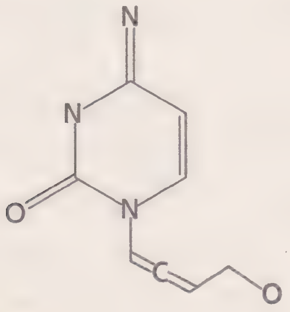


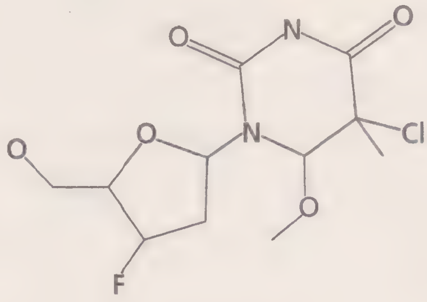
E = 9 E = –5 E = –5



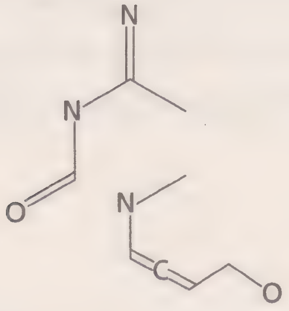
E = –5 E = –8 E = –8

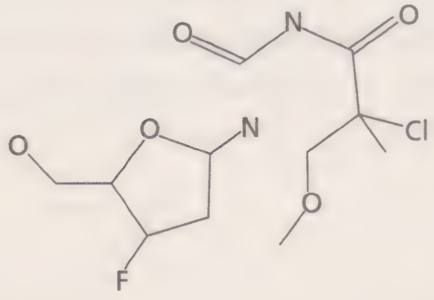
The next evolutionary stage is reproduction, in which we use the selected molecules to create new molecules that combine the properties of the parents. So, we cross the two molecules selected in the previous stage to generate two new molecules, which will be crosses between their two parents:



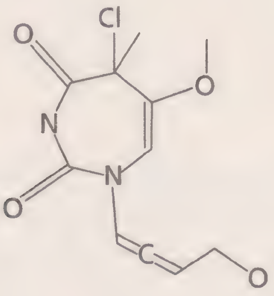


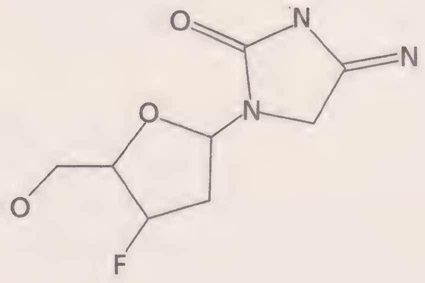
In the next illustration we see how the two molecules are divided into two segments:



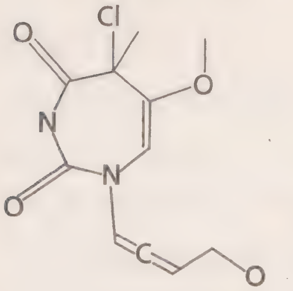


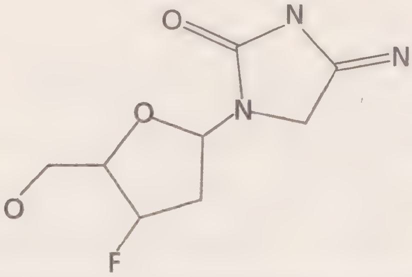
Ultimately, two new molecules are born that combine the segments of the previous molecules:

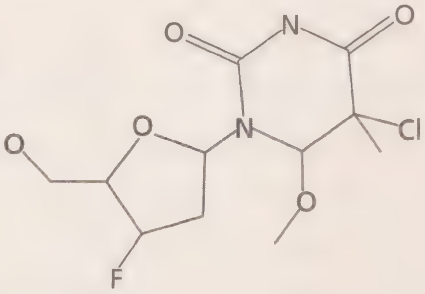




Finally, at the replacement stage, the individuals from the first generation are removed and replaced by the new individuals created. The most widely used replacement method, and the most simple, is elitism, in which all the molecules in a generation are replaced, except the best. In this case, the new generation contains the two molecules generated from the crossover, plus the best molecule from the previous generation, which will be the one with an interaction energy of –8.







After replacement, the evolutionary cycle is closed and repeats as many times as necessary. This means that these second generation molecules will now be evaluated, then selected, etc., and a third generation will be produced. The process continues until a predefined number of generations is repeated or the population converges, meaning that 90% of the individuals are the same molecule.

Of course, the reality is rather more complicated; our explanation here is a crude simplification, but isn't it beautiful?

Chapter 3

Machine Learning

Thursday, 6 May 2010, 9.30 am. The Wall Street stock markets open. To begin with, it is a normal day of financial transactions; the morning passes off without any obvious abnormalities. But in the afternoon, at 2.45 pm, for no obvious reason, some of the most important market stocks begin to collapse in value in a matter of seconds. Even given the volatility that was characteristic of the markets during that period of financial instability, this fall is very surprising, as some of the biggest and most robust companies suffer slides of over 60%, and within minutes the whole American stock market, and consequently the world stock market, crumbles.

That day, the Dow Jones Industrial Average (one of the most widely used benchmark stock indexes in the world) fell 9.2%, the biggest single-day fall in history, although it subsequently stabilised to close down 'just' 3.2%. In seconds a trillion dollars in value was wiped off the market, in what is now known as the 'Flash Crash'.

NYSE
NYSE
NYSE

Some of the first signs of the impending Flash Crash were detected on the trading floor of the New York Stock Exchange, on Wall Street.

While many reasons have been given to explain it, no clear cause has been identified. However, one of the hypotheses to have found most favour among financial investigators is the impact of HFT (High Frequency Traders), although this explanation has always been denied by market regulators. HFT are automatic, intelligent share and financial product purchase and sale systems capable of making decisions and acting in a matter of microseconds. It is calculated that nowadays 50% of international trades are conducted by HFT systems.

But, how can a computer system, intelligent or otherwise, make decisions of such magnitude so quickly? Any amateur investor will know that prices on the financial markets depend on a whole host of structural and short-term social, economic and political variables, ranging from the latest statements by the Finnish finance minister on labour regulations in her country to an unexpected fall in demand for crude oil caused by a rise in temperatures in southern Germany. How then could a computer system take account of so much information in order to make apparently intelligent share sale and purchase decisions and, what's more, make them in seconds? That is the question. Machine learning is one of the main pillars of artificial intelligence. We might not be aware of it, but a large proportion of the activities and scenarios we find ourselves involved in on a daily basis are completely controlled by intelligent machines. However, before they start to operate, they have to learn how.

An example of learning: predicting cancers

Cancer prediction is one of those cases in which artificial intelligence can be very useful to medical experts when it comes to making decisions at various diagnostic stages. Having a mammogram is, or should be, a regular practice for adult women in order to predict breast cancer at an early stage. A mammogram is just an X-ray of the mammary gland which highlights a number of anomalies in the breast, some of which may be incipient breast tumours. As a result, every time a radiologist identifies an anomaly in a mammogram, she orders a more exhaustive analysis, which requires a biopsy, or the extraction of tissue, a much more aggressive, uncomfortable and expensive process than the mammogram.

However, once the results of the biopsy have been analysed, a false positive is returned in 10% of cases. This means that although an anomaly was identified in the mammogram, the biopsy revealed no trace of a tumour. Therefore it would be very valuable to have a tool that could reduce this 10% of false positives as far as possible, not only to save public health costs, but also to avoid patients having to undergo the discomfort caused by aggressive biopsies and to limit their stress and anxiety at finding themselves in this situation.

On the other hand, there are also false negatives, when the mammogram does not show any anomaly but, unfortunately, there is a cancer. The health problems linked to the existence of false negatives are obvious, hence the importance of new diagnostic tools that can effectively reduce both false positives and false negatives. As we will see next, it is much more difficult to devise a tool that is capable of reducing false negatives than false positives. And false negatives are much more serious.

Imagine that an oncologist has to analyse a patient's mammogram to determine whether there is any sign of a cancer. Generally speaking, the reasoning or methodology that she follows can be broken down into the following steps:

1. Examines the mammogram and detects the most relevant characteristics in order to determine the new problem. The set of detected characteristics provide a picture of the situation.

2. Searches for previously diagnosed mammograms, both in her archives and the medicine books, that share similar characteristics.

3. Makes a diagnosis taking into account the diagnoses of the set of similar mammograms she has compiled.

4. Finally, if she thinks it is necessary, asks for a second opinion from colleagues in order to confirm her diagnosis.

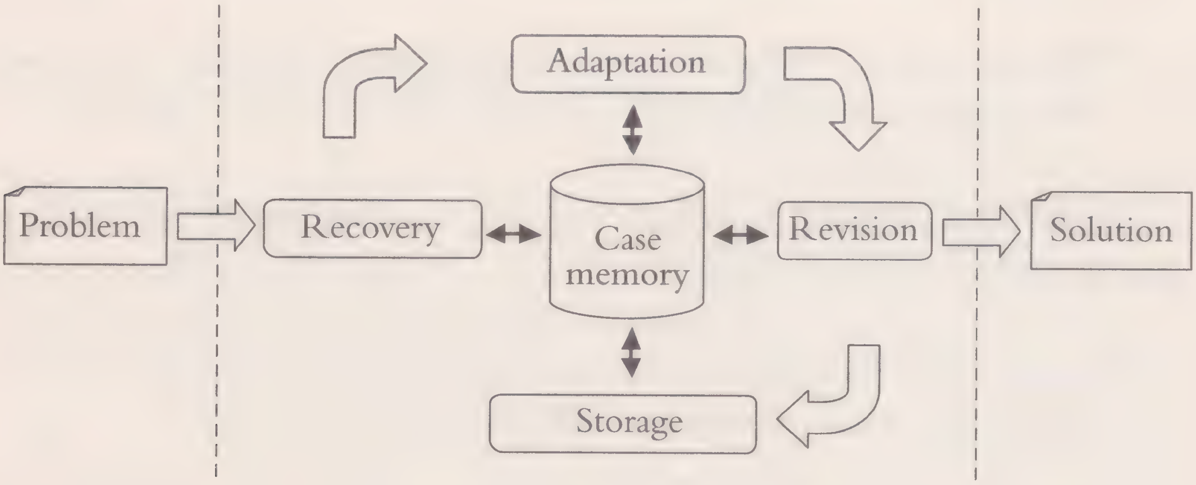
5. Based on the diagnosis made, makes relevant notes to summarise the new resolved case, as this may be useful in the future.

This procedure matches up at each stage with one of the most widely used prediction techniques in artificial intelligence, called Case-Based Reasoning (CBR). It solves new problems by looking for analogies with problems solved satisfactorily in the past and, once the most similar solution is selected, adapting it to the characteristics of the new problem. Thus CBR is not just a data analysis tool; instead, using the data analysed, it aims to achieve a more general aim: intelligent problem solving.

CHARACTERISTICS USED IN BREAST CANCER DETECTION

CBR, like other intelligent techniques, can be used to help diagnose the presence of malign tumours based on mammograms. As the input data in any of these techniques are numerical, an intermediate stage is required in which these values are extracted automatically from the medical images. In the case of breast cancers, various measurements are usually taken of common features of breasts called microcalcifications, which are tiny calcium deposits in the breast. Some of the characteristics generally used to detect malign microcalcifications in breasts are: area, perimeter, compressibility (ratio of area to perimeter), the number of holes they present, roughness (ratio of perimeter to irregularity), length, width, elongation (a ratio of width to length) and the position of the microcalcification's centre of gravity.

In the same way as an expert would store her experience in her memory or her notes, CBR has a 'case memory' data structure in which it stores previously solved cases. The following diagram illustrates how CBR works:



The first phase of CBR, retrieval, searches for solutions which most closely resemble the new problem in the system's case memory. When applied to our example, the aim of the retrieval phase is to search for diagnosed mammograms that share similar characteristics with the new mammogram to be diagnosed.

Next comes the reuse phase, in which the system tries to adapt the similar solution found to the characteristics of the new case. For example, suppose a logistics company needs to get a lorry from Lisbon to Rome and wants to use CBR to optimise the route. The first thing it will do will be to search its case memory for the most similar journeys made in the past. Imagine it finds another journey with an optimised route between Madrid and Milan. A large part of the route can thus be reused, and it only needs to optimise the route between Lisbon and Madrid, and between Milan and Rome. These two optimisations of a small part of the route can be performed using other conventional computing techniques, but the point is that this process of adapting the Madrid-Milan route to the new route that was set out, Lisbon-Rome, is part of the reuse phase.

Next comes the revision phase, in which the expert has to review the machine's diagnosis. It is in this phase that the person and the machine work together, making daily improvements to the machine's performance and, above all, increasing the reliability of systems in which prediction is critical. In the specific case of cancer prediction, given the great importance of the subject, it is very difficult for a healthcare system to agree to hand over completely the task of diagnosis over to an automated tool with these characteristics without the participation of a medical expert. However, who knows whether this might change in the future…

Finally, the last stage of CBR is the one in which, once resolved and reviewed by the expert, the decision is made whether the case will be retained in the case memory; in other words, whether it is sufficiently representative to be included in the set of mammograms which will be used to diagnose tumours in the future.

The success of CBR (but also of an expert's reasoning) depends on the ability to correctly implement each of the four phases of the method. Therefore, the four basic aspects that have to be taken into account during each phase are:

— Retrieval criterion: not all experiences are useful. The key is to determine which examples from past experience are selected for use when solving a new case. Accordingly, metrics or mathematical distances need to be defined that estimate what divides the new case from those stored in the case memory. For example, in the case of mammograms, given a new case to solve these mathematical metrics are used to determine which past, diagnosed mammogram most closely resembles the new mammogram for diagnosis.

- Reliability criterion: each field has its own implicit complexity and a specific level of risk, depending on the 'price to pay' for getting things wrong. In the case of cancer detection, the price to pay for producing a false negative is obviously much higher than for predicting that an anomaly is dangerous when it was really benign. Therefore, it is vital to establish mechanisms for the definition of criteria that help to ensure the proposal is reliable.
- Validation criterion: an expert is needed to validate the proposal. Because mammograms are so critically important, the validation phase is ordinarily performed by the expert radiologist.
- Knowledge maintenance criterion: the ability to solve problems is closely linked to the experience available. The consistency of this knowledge thus needs to be ensured, both by including new solved cases and eliminating those that confuse the system.

All these criteria have the same common denominator – the system's experience stored in the case memory. Ideally, the case memory will always have the following properties:

- Compact: it shouldn't contain redundant cases or noise because they can distort reality and confuse the system in the process of retrieving the most similar cases.
- Representative: we can't solve something we don't have evidence of; the system therefore needs cases that are representative of all the different characteristic aspects of the field so that it does not have a partial view of reality.
- Small: the system's response speed depends on the number of elements in its memory. The size of the memory has to be such that the system can respond in a reasonable amount of time.

These three properties can be summarised in the following premise: have a set of representative cases that is limited yet still representative of the whole field.

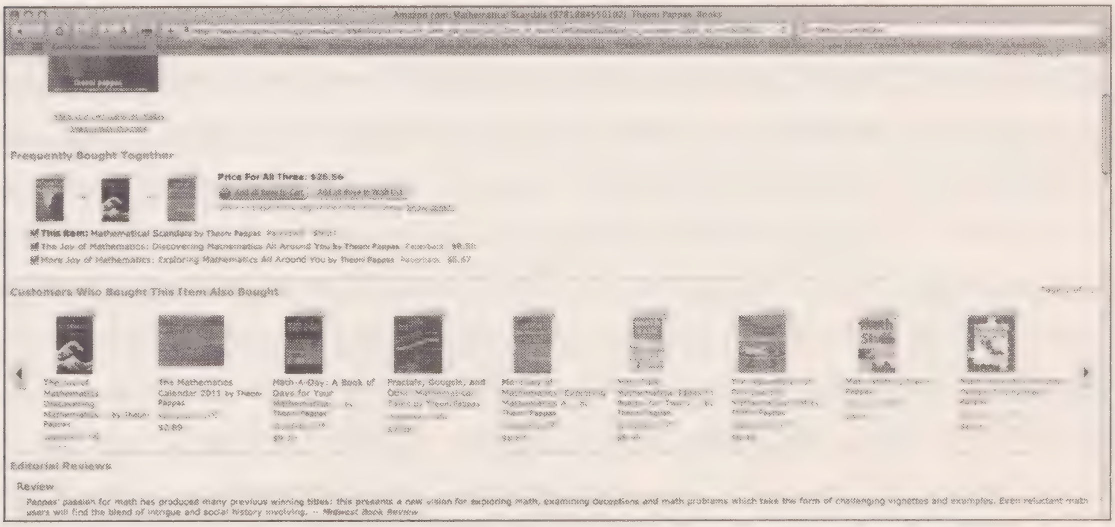
Another example: online marketing

Since the internet reached a mass audience, the discipline of marketing has changed completely. In fact, a very high percentage of everyday marketing is no longer static and is instead a more personalised form of mass marketing. For example, when we visit a web page or browse the web, adverts or *banners* start appearing at the top and sides of the pages we visit. These adverts are in no way random or static; instead a range of tools are used to monitor and analyse the surfer's behavioural patterns, and the adverts he sees are completely tailored in response to his current interests.

Who hasn't received an email in Gmail, Google's email service, and noticed that an advert appears alongside it related to the content of the message? And who hasn't visited a web page to look at something and then noticed adverts for hotels in Paris, a week after searching for exactly that?

All the mechanisms used by Google, and other similar companies, to target online marketing are intelligent tools that can instantly and automatically make advertising decisions without human intervention. In fact, if some type of human intervention were incorporated it would be impossible to perform so many marketing actions per second, as the number of web pages visited per second worldwide must number in the tens of millions.

If asked to name the most intelligent online marketing tool from all the tools available, many people would opt for the book suggestion algorithm used by Amazon. In fact, the same algorithm is used by other companies to drive similar offerings, such as Yahoo for its Radio LAUNCHcast, which takes the songs the user has rated positively to create his or her profile and then selects songs that other users with similar profiles have listened to and rated highly. On Amazon this system is clearly visible every time users search for something, whether or not they are registered users, in the section *Customers who bought this item also bought*.... Although it may

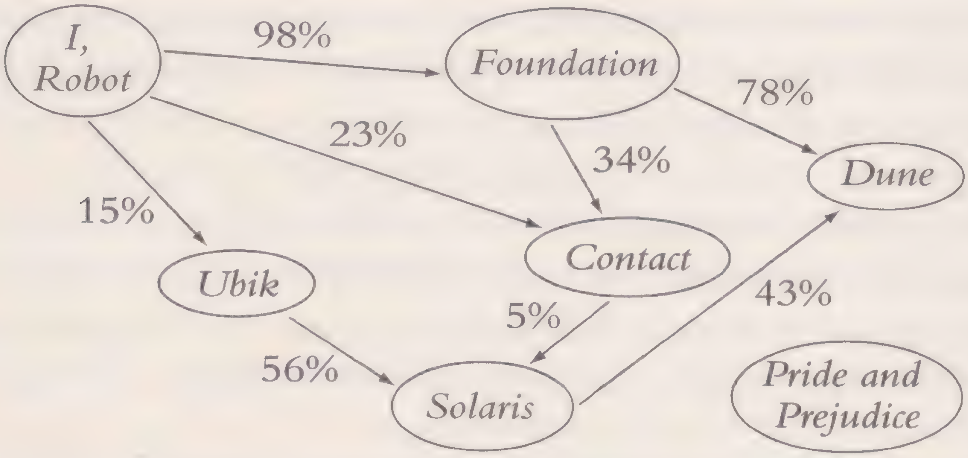

Frequently Bought Together
Customers Who Bought This Item Also Bought
Editorial Reviews
Review

Amazon's related search page.

seem straightforward, this very simple idea is underpinned by a highly complex system. It is actually based on techniques classed as artificially intelligent which do far more than merely explore the contents of the shopping baskets of other users who bought the same item that this user is looking at now.

The standard tool used to approach this type of problem is known as a 'Bayesian network'. In fact, the largest global research centre specialising in this tool is the Microsoft Research Institute, which is examining the application of this technique not just to online marketing but also to other areas, so that the Windows user interface automatically adapts to each individual's working style or preferences.

Underpinning a Bayesian network is the idea that there is a chain of events that usually happen and can share probabilities with other chains of events; that is why they are called 'networks', because they are chains of intertwined probabilities. Take the example of book purchases:



In the network in the example we can see that 98% of customers who bought *I, Robot* also bought *Foundation*. On the other hand, no customers who bought *Dune* also bought *Pride and Prejudice*, therefore no relationship is established between these two books. So, if the system detects that a customer has just bought *I, Robot* and is now searching for information on *Foundation*, the recommendations section will now display *Dune* and *Contact*, as a very significant percentage of people who bought the two former books also bought the latter two. All this amounts to an automated marketing campaign tailored to each customer which the system uses to try to boost sales by advertising two products that the customer perhaps didn't know existed, but as the system has plenty of information on other previous buyers, it has been able to establish this network of causal relationships and is using it to make new recommendations.

OTHER USES OF AUTOMATED MARKETING

Interior of a New York supermarket (photo credit: David Shankbone).

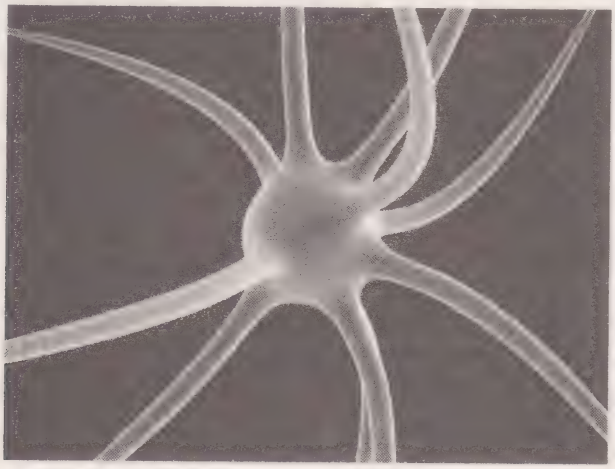
Marketing in supermarkets is designed to ascertain our tastes and needs to make it easier for us to fill our shopping trolleys. Automated marketing applies not only to the virtual online world: nowadays, banks, telecoms operators and even neighbourhood supermarkets use this new approach. For example, we're all familiar with the discount coupons we get from the supermarket where we do our weekly shop. Naturally, they don't usually give us discount coupons for the products we buy routinely already (supposing, of course, that they do it properly). Instead we get coupons for products we don't usually buy but other customers with similar shopping baskets to ours do buy. This brings the product to our attention, so that having perhaps not been aware of it before or ever bought it, it might from now on become a regular in our baskets. The same thing happens with other types of companies, like financial services or telecommunications companies, which regularly send us offers for products we were not aware of but which the companies infer might interest us, based on our customer profile.

Similarly, the system also knows that advertising *Pride and Prejudice* to a customer who is buying science fiction, which is what would happen in a conventional marketing campaign, is a waste of time. In a traditional marketing campaign, the new edition of *Pride and Prejudice* would be advertised during a cultural review programme broadcast at 11pm on a minor channel, for example.

However, while media buyers might choose to place an advert in this programme and time slot when viewers most likely to be interested in the product would be watching, there would still be a large number of viewers who liked science fiction books on whom this advert would have no effect, with the financial loss that this entails for the advertiser. For a static marketing channel, like television, radio or billboards, it is impossible for the advertiser to know the individual profile of customers at any one time and, if it did, it lacks the tools needed to adapt the advert carried to each of them.

The robot's brain: neural networks

Robotics is one of the most complex areas of engineering, not only because of all the electromechanics and servo control that need to be deployed in a simple robotic arm, but also because of the sophisticated mathematical calculations required to calculate the trajectories of moving parts. To achieve this, in some cases robots have artificial brains made up – like our brains – of neural networks. In this case, however, the neurons are artificial.



Computer graphic of one of the neurons that make up the human brain (source: Nicolas P. Rougier).

The concepts of 'neural networks' and 'artificial neurons' have gone through various cycles of euphoria and disappointment in the course of their short lives. Their birth can be traced back to the 1940s, when Warren McCulloch and Walter Pitts proposed the hugely successful Threshold Logic Unit. An artificial neuron is the encapsulation of this algorithm, which is defined by computing professionals as:

$$\text{Input}_1 \rightarrow X_1$$
$$\text{Input}_2 \rightarrow X_2$$
$$\ldots$$
$$\text{Input}_i \rightarrow X_i$$

$$\text{if } \Sigma\ (X_x \cdot \text{Weight}_i) > \text{Threshold},$$
$$\text{then Output} \leftarrow 1$$
$$\text{otherwise Output} \leftarrow 0$$

which means, in everyday language, that if the stimulus [the sum of products (X_i x Weight$_i$)] exceeds a determined threshold, then and only then the neuron fires. As you can see, a neuron is an extremely simple element, as it only implements a small number of arithmetical operations and a comparison. This fact facilitated the implementation of artificial neurons in microchips, so that full neural networks could be implemented in hardware from the end of the 1990s. These microchips are now used to build electronic prediction devices, such as instruments capable of detecting the reason why a crying baby is uncomfortable.

An artificial neuron functions in much the same way as a natural neuron and, as we have seen, it works in quite a simple way. In fact, the difficulty with neural networks stems mainly from two elements that have to be adjusted and on which the network's ability to make reasonably accurate predictions depends: the weight of the various inputs and the threshold. The difficult process of adjusting these values so that, given a series of inputs, the neuron can produce the desired output is what is known as the 'training process', or in psychological terms, 'learning'. The breakthrough in neural learning was made in the late 1950s when Frank Rosenblatt invented a neuron that could adjust weights and thresholds called a 'perceptron'.

In biological terms, a natural neuron behaves in practically the same way. Each neuron has a set of inputs through which it perceives electrical signals sent by other neurons, known as 'synaptic connections', and based on these it evaluates whether these stimuli exceed a sensitivity threshold, bearing in mind some synaptic connections are more important than others (the weights we discussed before). If this sensitivity threshold is exceeded, an electrical signal is sent along the axon, or its equivalent in an artificial neuron, the output.

Because of its relative simplicity, the perceptron became a useful predictive tool. Given a data point, it predicts whether it belongs to one category (0) or another (1). A classic example is the problem of the *Iris* botanical genus, three species of which are *Iris setosa*, *Iris versicolor* and *Iris virginica*. Each sample gathered is defined by four parameters: the length and breadth of its petals and sepals. The aim is that the tool can tell which species a new sample belongs to. In this case, we use three perceptrons, each one specialising in detecting just one of the three species, so that if the new sample is of the species *Iris setosa*, just one of the perceptrons should return the value 1, and the other two, 0.

Iris setosa, I. versicolor *and* I. virginica*, in that order. Depending on the shape and dimension of the petals and sepals, the system is able to categorise new flowers.*

Readers might now be wondering why statistical tools are not used to solve the *Iris* problem. Indeed, this problem is so simple that it could also be solved using classic statistical tools, such as principal component analysis. However, it is worth noting that the perceptron and statistical tools use two very distinct reasoning methods, and the perceptron's method could be said to be closer to natural reasoning.

If we used statistical tools to solve the *Iris* problem, we would produce rules like: "If the petals are between such and such a length, and their width is between such and such a value, it is likely that this sample belongs to species X." The perceptron, meanwhile, reasons as follows: "If the petals are between such and such a length, and their width is between such and such a value, it is likely that this sample belongs to species X, unless its sepals are so short that it does not matter what the petals are like because I can state that it is species Y."

This means that when weighing up the value of the inputs in order to make a decision, some items of information are much more significant than others, but if an extreme value is reached, an input that was previously unimportant becomes very relevant to the decision-making process.

Neurons cluster together

Despite the highly innovative nature of the perceptron and the wide range of applications that it had, it soon became clear that it could not be applied to a certain group of 'non-linearly separable' problems. Unfortunately, most real-life problems belong to this category. As a result, in the 1980s many criticisms were made of neural networks as part of a debate that often went beyond the scientific sphere and were designed to personally discredit defenders of the perceptron.

This coincided, to the greater disappointment and frustration of scientists researching these areas, with the dark ages of artificial intelligence, a period characterised by very significant cuts to research budgets in the US and Europe. Firstly, society came to realise that the idea advanced in films like *2001: A Space Odyssey* would not come true for a very long time; secondly American government agencies, which had had great hopes that artificial intelligence would tip the balance of the Cold War in their favour, suffered major setbacks, including in the field of machine translation, which was hugely important for interpreting Russian technical documents. Despite the massive cuts in funding resulting from the discovery that the perceptron could not tackle non-linearly separable problems, research into the topic continued, although at a much lower intensity and sometimes in secret, to avoid researchers being ridiculed by the far greater number of detractors. But, could the problem of non-linear separability actually be settled?

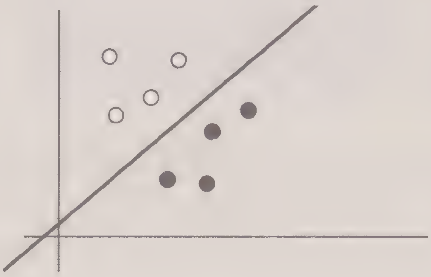
The answer came in the late 1980s and was so obvious and logical that it was difficult to understand how researchers hadn't noticed it before. The natural world itself had already discovered the answer millions of years before. The solution lay in connecting several perceptrons up to form what is known as a 'neural network'.

The diagram overleaf shows a three-layer neural network, with an input layer, hidden layer and output layer. This neural network is called a 'feed forward' network,

For many years people thought that the creation of superintelligent computers like HAL 9000, from 2001: A Space Odyssey, *was a very real possibility. They were soon disappointed.*

LINEAR NON-SEPARABILITY

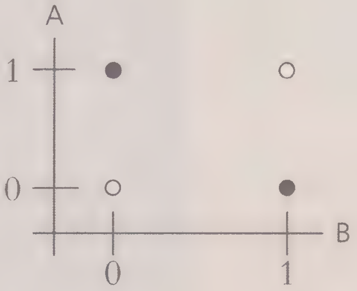
If we consider a situation in which data may belong to two categories and each of these is described by two descriptors (therefore, two inputs), we could draw a graph like the following, with eight samples collated.



In it, the white dots represent the data in category A, and the black dots the data in category B. As you can see, it is easy to draw a line separating the two categories, and this is exactly what a perceptron does when the thresholds and weights of each input are adjusted. However, what happens if we analyse the X-OR synthetic problem? X-OR is a logic operation (it means exclusive "or") which fulfils the following relationship:

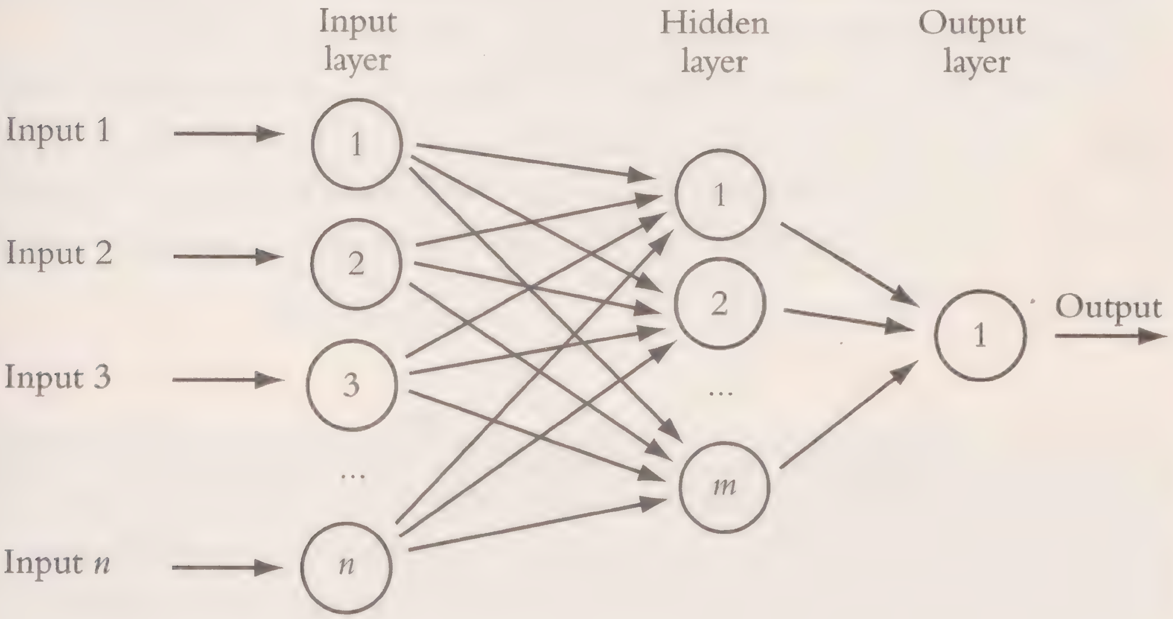
Inputs	Output
00	0
01	1
10	1
11	0

Now the graph looks like this:



In this case it is not possible to draw any straight line that separates the white dots from the black. We are therefore faced with a non-linearly separable problem. A perceptron could not be correctly trained to solve a logic problem as simple as X-OR.

as the data always flows from left to right and no cycles are formed between synaptic connections.



But a neural network can be as complex as the programmer wishes, with as many hidden layers as he thinks appropriate and, in addition, connections that can feed forwards and backwards to simulate a kind of memory. Indeed, neural networks have been constructed that comprise 300,000 neurons, the same number of neurons as there are in the nervous system of an earthworm.

In a neural network learning is complicated, and therefore engineers have devised a large number of learning methods. One of the simplest is back-propagation, which also gives its name to the neural networks that use it. This method consists in minimising the output error of the neural network by adjusting the input weights of the synaptic connections in the neurons from right to left using the gradient descent method. In other words, random values are first assigned to the weights of all connections in the network. A pattern is then introduced with a known value that has to be predicted. For this reason it is called a 'training pattern'. As you might expect, the result given by the output neurons is a random value. Based on this value, beginning with the neurons closest to the output and finishing with the input layer neurons, the connection weight values are adjusted so that the value of the output neuron approaches the actual known value.

This procedure is repeated hundreds or even thousands of times with all the training patterns, and when it is complete an epoch is said to have passed. Next, the process can be repeated for another whole epoch with the same patterns. A typical learning process comprises tens of epochs. This process is similar to the psychological learning process, in which people learn by repeatedly studying the same data.

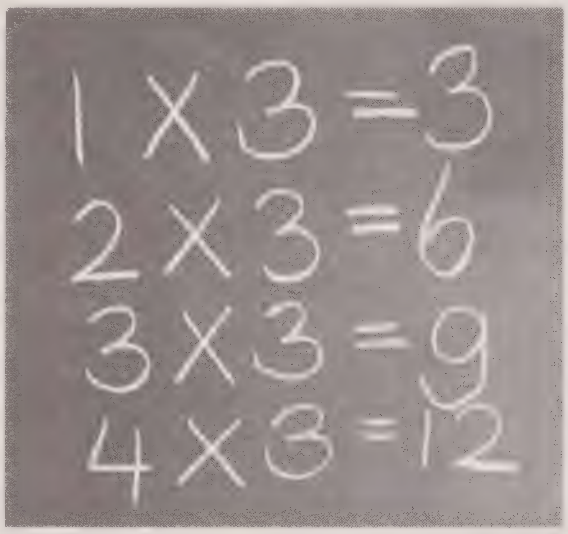
The architecture of neural networks and learning methods have gradually been refined over the years, and many different types of neural networks have emerged, with very varied real-world applications. One of the most widely used networks nowadays is the Hopfield network, which uses a form of memory system called 'associative memory'.

THE OVERTRAINING TRAP

A prediction system based on machine learning infers its predictions from the generalisations it is able to make from past cases. Therefore, when the system is incapable of generalising, it loses its usefulness.

When the training process is repeated too many times, there comes a point when the adjustment is so precise and so adapted to the training sets that the system, as it has memorised the sets, is no longer basing its predictions on a generalisation but rather on memorisation. When this happens, the system is now only capable of making correct predictions when provided with data from the training set. Whenever different data is input for it to make a prediction, the prediction will be incorrect. The system is then said to have been 'overtrained'.

In some ways, the same thing would happen to a child who, instead of learning to multiply, just memorised the tables. If he was asked for one of the sums he had memorised, he would undoubtedly give the correct answer, but he would be unable to solve a new multiplication not found in the tables.



Multiplication tables are a good example of learning by rote.

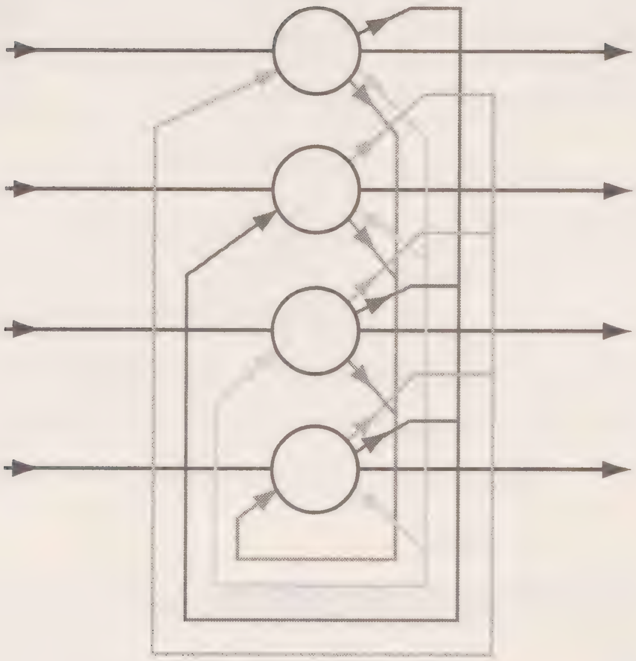


Diagram of a Hopfield neural network.

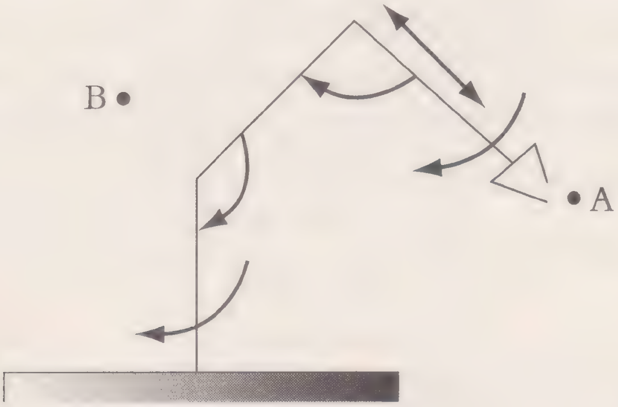
In associative memory, information is ordered according to content and, therefore, to access it we have to indicate which content we want to access, instead of giving a physical/electronic location, as happens with a computer's hard drive or RAM.

Other commonly used types of neural networks are self-organising networks or maps and Kohonen maps. This type of neural network features an innovation: learning is unsupervised; instead the network self-learns from its own errors as it works.

And so the brain functions

Inverse kinematics is a discipline of physics that seeks to calculate the chain of movements needed to move an object from a point in space A to another point B. These calculations become exponentially more complicated, in terms of the number of matrix movements that have to be resolved, as degrees of freedom are added to the system.

For example, consider the robotic arm with four joint angles and a retractable extension shown in the following diagram. If we wanted to solve the inverse kinematics matrix equations conventionally, a supercomputer might take hours to perform all the calculations needed to find out by how much and in which direction each degree of freedom needs to be changed in order to move the tool at the end of the arm from A (starting point) to B (end point).



It is thus impossible to support robotic systems that modify their trajectories in real time by solving matrices in a conventional way. In the case of robots that systematically perform the same repetitive tasks (think of the robots on a car production line), it is in principle possible to calculate and program, step by step, all the movements that the motors in each of the robot's joint angles or retractable arms have to perform.

However, if we wanted to deploy a robotic arm capable of acting independently and with the ability to coordinate its actions in response to a concrete situation (think of the robots on spaceships, those used in surgery or, simply, the first experimental robots for domestic use), we would need more innovative systems that enable the robot's processing components to quickly calculate how it needs to move to accomplish its task.

To achieve this, we use one of the most efficient motor control systems in existence today: the back-propagation neural network. In the case of the robot, the neural network trained to control it will have as many outputs as degrees of freedom available to the robot, and each one of them will indicate the magnitude and direction in which each motor has to move to go from the starting point to the end point.

The major drawback of this approach is that the neural network has to go through a long learning process, which is not necessary in a conventional approach. In some ways, this could be compared to the learning process experienced by human beings, who as children learn to walk based on trial and error; however, once they have mastered walking, they can take steps without solving the complex physical equations that the kinematics of their legs solve in order to move and not lose balance.

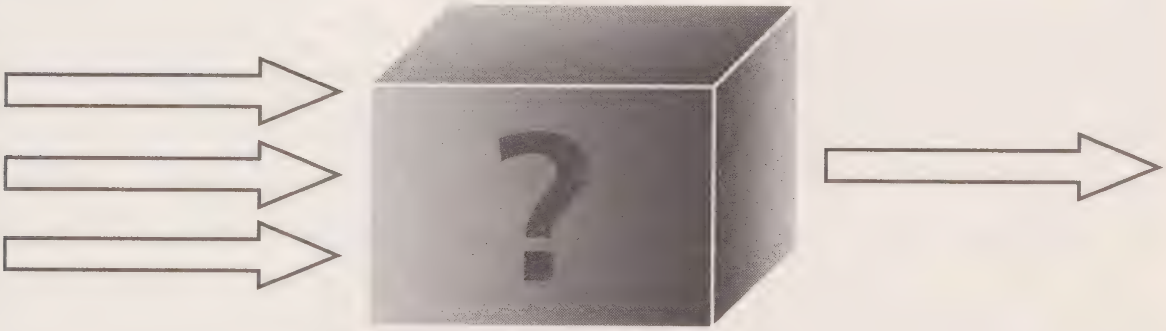
Thus, the usual training method used in these instances is to show the back-propagation neural network hundreds of thousands of possible trajectories, tens of thousands of times, and teach it how the different motors move in each case in order to go from the starting point to the end point.

Once the network is properly trained, it is said to have learned the sensory-motor map. This means that a robot's central processor can precisely solve, to the millimetre, the inverse kinematics of the robot in mere milliseconds.

The brain grows ever more complicated

The good results produced by neural networks led, in the 21st century, to their standardisation as the go-to tool for solving many problems. However, major weaknesses remained.

The first of these was overtraining; it is quite easy to fall into this trap when training a neural network. The second was the large number of parameters that have to be adjusted arbitrarily 'by hand' before going on to the stage of training the neural network. The problem with adjusting all these important factors by hand is that there is no manual or methodology for doing the job. So massive human and



The neural network makes predictions, but it is unclear what reasoning process it follows to produce them. Some people compare it to a crystal ball.

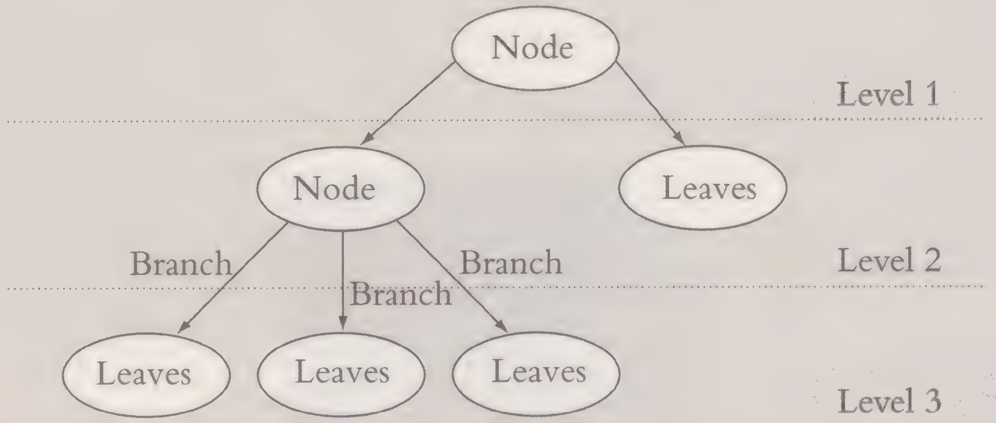
technical resources have to be invested (in most cases using the old, despised method of trial and error) in order to adjust all these characteristics. A third problem, more philosophical than practical this time, is that we don't fully understand how a neural network reasons once it has been trained.

This was not regarded as important until neural networks began to be fully deployed to solve real problems. For example, if a neural network is used to control a car's anti-lock braking system (ABS), the engineers will naturally want to check that they understand, in the finest detail, how the network reasons, so that they can be sure the brakes won't fail in any of the thousands of different braking situations they might face.

For these important reasons, up to the end of the 1990s computing theorists worked hard to design new computational methods that would resolve or mitigate these effects. The solution was eventually delivered in the early 21st century by Vladimir Vapnik and his team from the famous telecommunications and

THE INFORMATION TREE

The tree is a type of information organisation structure which is very widely used in engineering, as it makes it possible to arrange and link data hierarchically. As a data structure, the tree uses a specific nomenclature, which is worth knowing:



Each one of the data items in a tree is called a 'node', and these nodes, which represent a unit of information, are divided into different levels and are interlinked by branches. A node linked to another on a different level will be its parent if it is of a higher level, or conversely its child. Lastly, nodes without children are called 'leaves'.

Will this classroom image become a thing of the past?
Many pupils would certainly be happy if it did...

electronics company AT&T Bell Labs. Vapnik devised Support Vector Machines (SVM) based essentially on the idea that by adding new artificial dimensions to a non-linearly separable problem it would, thanks to its new artificial dimensions, become linearly separable.

SVMs have successfully overcome most of the drawbacks that had become apparent in neural networks (above all overtraining, the setting of initial parameters and the incomprehensibility of the network's reasoning), and have now replaced them in virtually all fields of computing. However, neural networks are still used in some industrial applications, such as in the field of robotics, because they are so simple to implement in hardware.

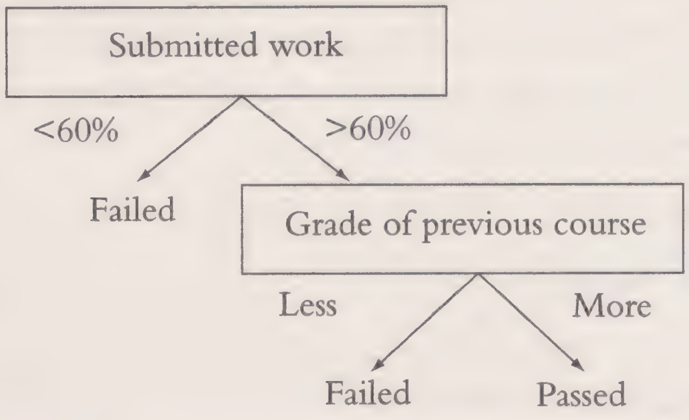
Are exams necessary?

Can artificial learning be applied to natural learning? As we have seen, artificial learning can be applied to all fields of science and technology, but is it possible to go further and apply it to the social sciences, education in particular? How does a teacher determine the level of knowledge her pupils have reached? Could some of the subjective criteria often used by professors and teachers to assess their pupils be automated? And can we predict a pupil's performance without having to examine him? All of this could be done using a technique as simple as decision trees.

Decision trees are a very simple yet very effective pattern recognition tool. A decision tree seeks to determine which are the most decisive or discriminating variables that will correctly classify members of the population. Take the example of the prediction of pupils' grades at a given institution. We have the following training data:

Grade from previous year	Attendance in class	Work submitted	Objective: pass or fail?
Higher than average	100%	45%	Fail
Higher than average	100%	100%	Pass
Higher than average	90%	100%	Pass
Lower than average	85%	30%	Fail
Lower than average	100%	80%	Pass
Lower than average	99%	100%	Fail
Lower than average	100%	55%	Fail

A decision tree that successfully models these data might be the following:



In this case, attendance in class is not a discriminating variable, as it is not one of the nodes in the tree. There are various methodologies for determining whether the variable in a model is discriminating. One of the most widely used methodologies is Shannon's entropy. This method analyses, at each level of the tree, which variable generates the lowest entropy, and this variable is selected to discriminate at this level of the tree. Let's look at this in more detail.

Shannon's entropy, S, is defined by the following formula:

$$S = -\sum_{i=0}^{n} n_i \cdot \ln(n_i).$$

Let's look at it in action in the exams example. At the first level of the tree we need to analyse the entropy generated by each variable. The first of these is the variable 'grade in previous year'. If we separate the sets for this variable we are left with two subsets of data, one where

$$S_{\text{Grade in the previous year lower than average}} = -0.75 \cdot \ln(0.75) - 0.25 \cdot \ln(0.25) = 0.56,$$

as of the pupils who last year had a grade lower than the average, 75% failed and 25% passed, another where

$$S_{\text{Grade in the previous year higher than average}} = -0.33 \cdot \ln(0.33) - 0.67 \cdot \ln(0.67) = 0.64,$$

as a third of pupils who last year had a grade higher than the average failed and two-thirds passed.

This operation is repeated for each variable. The next variable is 'attendance in class', in which, to simplify, we distinguish between higher than 95% and lower than 95%. In this case,

$$S_{\text{Class attendance greater than 95\%}} = -0.6 \cdot \ln(0.6) - 0.4 \cdot \ln(0.4) = 0.67;$$
$$S_{\text{Class attendance less than 95\%}} = -0.5 \cdot \ln(0.5) - 0.5 \cdot \ln(0.5) = 0.69.$$

Finally, we analyse the variable 'work submitted' (or exercises handed in), where, to simplify again, we distinguish between higher than 60% and lower than 60%. Like this:

$$S_{\text{Submitted work greater than 60\%}} = -0.75 \cdot \ln(0.75) - 0.25 \cdot \ln(0.25) = 0.56$$

and

$$S_{\text{Submitted work less than 60\%}} = -1 \cdot \ln(1) = 0.$$

Therefore, the most discriminating variable is the last one, as the entropies of the subsets that it generates are 0.56 and 0.

In this case, the training sets that fall within the 'lower than 60% of exercises handed in' category are all fails, and therefore we no longer have to worry about this branch of the tree. However, the other branch contains as many fails as passes; as a result, we have to continue the analysis recursively, without taking account of the sets that have already been discriminated.

We are now only left with two possible decision variables: 'grade in previous year' and 'attendance in class'. The Shannon entropies for the groups generated using the former discriminating variable are the following:

$$S_{\text{Grade in the previous year lower than average}} = -0.5 \cdot \ln(0.5) - 0.5 \cdot \ln(0.5) = 0.69;$$

$$S_{\text{Grade in the previous year higher than average}} = -1 \cdot \ln(1) = 0,$$

while if we analyse the behaviour using 'attendance in class', we get:

$$S_{\text{Class attendance greater than 95\%}} = -0.33 \cdot \ln(0.33) - 0.67 \cdot \ln(0.67) = 0.64;$$

Therefore we select the discriminating variable 'attendance in class', as its entropies are lower.

The method for constructing decision trees and, therefore, the method that ensures that the trees learn, is simple and elegant, but has two major disadvantages. The first is that in problems with a large number of decision variables it runs very slowly; and the second, more serious, problem is that it can very easily fall into local optima. In other words, as the tree is never analysed in its entirety, but rather on a level-by-level basis, it is possible that a given decision variable may minimise the entropy at a given level and be selected, but conversely if another variable was selected, the tree would classify the population better overall.

One 'trick' often used to improve the success rates of decision trees is to use 'forests', i.e. to train various trees, each using a different method, and produce the final prediction on the basis of the consensus of the predictions made by each tree in the forest.

Using this philosophy, the most frequently used methodology for training a forest is to construct decision trees by randomly selecting decision variables; this means that if we want to train a forest of 100 trees, for each tree five random decision variables are selected, and the tree is trained using only those five variables. This approach is known, poetically, as the 'random forest' method.

Chapter 4

Automated Planning and Reasoning

The following events could occur on any given day:

2.32 pm:	A lorry travelling too fast overturns on a minor road. The driver suffers a heavy blow to the head.
2.53 pm:	An ambulance and fire engine arrive at the scene of the accident and in a few short minutes manage to pull the driver out; he is unconscious and has a severely fractured skull.
3.09 pm:	The ambulance arrives at the hospital, where the emergency department announce that the driver is brain dead.
3.28 pm:	The patient is identified and his family are informed.
4.31 pm:	When they arrive at the hospital, a team of psychologists contacts the dead man's family to give them emotional support and get their consent to donate those organs unaffected by trauma.
4.36 pm:	After a brief discussion, the family agree to donate the kidneys of their dead family member (henceforth, the donor).
4.48 pm:	A surgical team begins work on removing the kidneys and checking their medical condition. At the same time, hospital managers process the paperwork needed to get legal authorisation.
5.24 pm:	Having removed the kidneys, the donor's biological data and the characteristics of the organs are entered into an information system.

And so begins an organ transplant.

How a transplant is managed

5.24 pm:	The information system immediately identifies who the two recipients of the kidneys will be, informs them, and assigns and plans the logistical

resources needed to transport them. In one case an ambulance needs to be prepared to take one of the kidneys to the hospital in a neighbouring city (30 km away), and in the other a light medical transport plane will take the kidney to a city 450 km away which is part of a different health service region. The second organ is transported from the donor hospital to the nearest airport by helicopter, which the same information system assigns automatically. At the same time, this system is also undertaking many of the administrative formalities needed by both health systems, the donor hospital and the recipient hospital.

6.10 pm: The first transplant begins at the hospital in the neighbouring city.

7.03 pm: The second transplant begins in the city 450 km away.

9.00 pm: Both recipients are receiving the appropriate post-operative treatment and immunosuppressants, and both are making good progress.

This is a description of the Spanish organ transplant system, which is regarded as one of the best in the world and might soon be extended across the EU. What makes it such a success? As readers may have guessed, the Spanish transplant system is based on a powerful artificial intelligence system covering the country's whole hospital network, which not only examines and takes into account the needs and characteristics of each recipient and all the logistical details, but also the country's complex and disparate transplant rules.

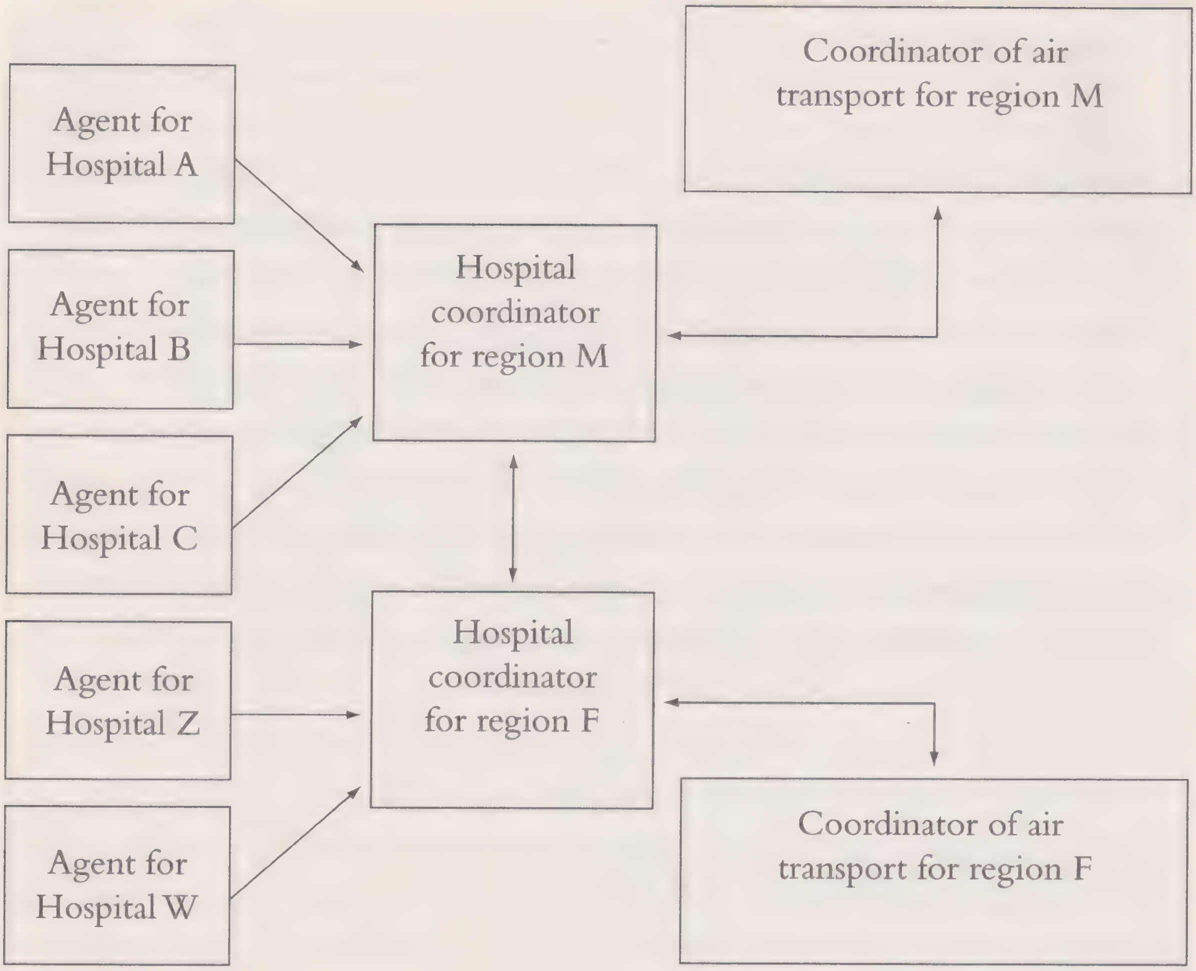
This intelligent system is based on a multi-agent system, which means that it is made up of various information systems. These are relatively simple but very specialist in the way they operate, and provide the whole system with a powerful collective intelligence that makes it the best transplant model in the world. A transplant coordination system usually has a multi-level structure, for example split into national, zonal, regional and hospital levels and, in the area of hospitals, recipient data can be stored across the hospital network or centrally in a 'data warehouse'. Because of all of this, there are a large number of intelligent agents that manage information on recipients and which are consulted continually by other intelligent agents activated whenever there is a donor. Other agents in the system take responsibility for various aspects, such as planning and assigning logistical resources for the critical organ transport step, and managing the administrative formalities required by the various regional health systems.

PIGEON RANKING

Google has a massive reputation in the world of computing and artificial intelligence. The algorithm Google uses to produce the rankings in the searches that internet users the world over carry out millions of times per second. There is so much interest in this algorithm and so much pressure on Google to publish it that on the morning of 1 April 2002 the company decided to put a link on its home page in which it said that it would explain its ranking algorithm. The algorithm in question was called 'PigeonRank', and its strange name was in no way picked at random. The article that Google published referred to dense systems of PCs ('pigeon clusters'), with each pigeon using a screen and a keyboard. The description of the algorithm claimed that every time a user performs a Google search, each of the web pages that is relevant to this search is shown to one of the pigeons, which then begins to peck at its keyboard. The web pages are then ranked according to the number of pecks by the corresponding pigeon.

In the same article, Google explained how it cared for the pigeons, how they were kept and how they were selected. It also claimed that other types of birds had been tested, including chickens and various species of birds of prey, but that pigeons proved to be the most intelligent and suited to the task. The article even dared to claim that, although it was true that no pigeon had taken up a seat on the Supreme Court, they had proved their skills as air traffic controllers and football referees.

Many rival software engineers were baffled until they double-checked the date.


Agent for Hospital A
Agent for Hospital B
Agent for Hospital C
Agent for Hospital Z
Agent for Hospital W
Hospital coordinator for region M
Hospital coordinator for region F
Coordinator of air transport for region M
Coordinator of air transport for region F

Simplified example of a network of agents responsible for coordinating organ transplants.

Working with an intelligent multi-agent architecture like this has many advantages, such as the fact that the system is fail-safe redundant. That means if an agent or set of agents fails, it can self-regulate and assign other agents to perform the tasks that need to be performed. Another major advantage, and one which is very clearly desirable in the case of organ transplants, is that by deploying relatively simple yet very specialised agents, an intelligent system can be constructed that can perform a variety of complex, multidisciplinary and critical tasks at the same time and in a matter of seconds.

AGENT-ORIENTED PROGRAMMING

Computer programming is a rapidly evolving field. There are now five main families of programming language, and object-oriented programming is the language most used by modern programmers. It is a type of programming in which everything is represented by units of information called 'objects'. These have a series of attributes that store information on these objects and are capable of performing various operations on the basis of this information. Object-oriented programming requires the intervention of coordinators that have intelligence and can ask the objects to perform one task or another, but always on the assumption that the object is basically an unintelligent item waiting to be told what to do. However, there have recently been rapid developments in this area of programming, and the emergence of agent-oriented programming. In this type of programming 'dumb' objects become agents with much more intelligence and autonomy, so the coordinator's work is not as critical.

Planning is the operative word

Planning and scheduling series of resources that vary in number so that a given task can be carried out successfully can be a highly complex activity, even for an experienced human. Similarly, planning is an aspect of all areas of real life, from non-critical tasks, such as planning the teachers, subjects, groups of students, classrooms, laboratories and audiovisual resources available to a secondary education establishment, to the critical planning of resources for putting out a forest fire or in an emergency resulting from any other natural disaster.

On the other hand, automatic reasoning is a very simple task for a human but a genuinely complex one for a machine. In fact, 'reasoning' is very much a defining human characteristic and the keys to our reasoning have yet to be clearly explained by neurobiologists and other experts in the field. With the aim of imitating this reasoning, engineers have developed a number of very interesting techniques that can be applied, for example, to forest fires.

Nowadays, various governments base their forest firefighting protocols on planning systems driven by artificial intelligence. Normally, when a medium-sized forest fire is declared, a firefighting expert takes an hour to an hour and a half to design a firefighting plan; this plan sets out the process to be followed depending on the resources available at the time, which in turn depend on a series of factors, including terrain and weather factors. However, one of the problems which these experts often face is that the conditions are changeable – and they can change faster than it takes to redesign the firefighting plan. As a result, many governments are trying to deploy automated systems that can devise these plans in a matter of seconds. To do this, the system collates parameters such as terrain, weather conditions, accesses available to the area affected by the fire, the availability of air or land-based resources, and coordination and communication with various control units and centres, and on that basis it designs a plan that is then reviewed by human experts.

A forest fire requires the coordination of many human and material resources.

It could be, for example, that at a given moment a ground unit becomes free and the system has to consider two options – move it to an area where the fire is very active or send it to put out the flames in another closer and less dangerous area. How can the system decide which of these two options is better? Naturally, the aim is to put the fire out and, therefore, it seems more plausible for the unit to be sent to the area where the fire is most virulent; on the other hand, it could take hours to get there, while in an area a few minutes away the flames are less dangerous and could be extinguished more easily, so perhaps the fire should be attacked there. The question is how can we clearly and objectively quantify, as a conventional non-intelligent planning system would need to do, the benefits of putting out the fire in an area, taking account of the distance to be covered and the time elapsed? Now suppose that we have ten or twenty times the number of ground-based units, in addition to air-based teams, and factor in variables like wind strength and predicted changes in it, rain, inhabited areas, environmentally sensitive areas, etc. You can begin to see just why an intelligent system is needed that can take decisions taking account of all these details and a range of parameters.

FUZZY LOGIC

Fuzzy logic is a type of mathematical logic that tries to approximate the logical methods and operations used when acting in a human and natural way. In real life, things are rarely black and white. However, in classical logic, like Boolean logic, values can only be true or false, so we are always obliged to adopt extreme positions.

For example, if I ask whether the goalkeeper of a Kazakh first division football team is good or bad, the answer may be "It depends". If he is compared to the cream of world goalkeepers, he very likely won't be that good, but if he is compared to my neighbourhood team's goalkeeper, he'll probably seem like an excellent player. So, fuzzy logic variables do not have the values 'true' or 'false', but rather a real value, normally between 0 and 1. If the value is '1' it is 'true' and if it is '0' it is 'false'. Pursuing the football analogy, if we qualify 0 as being absolutely useless at stopping any type of shot, and 1 as the excellence of the best goalkeeper in the world, the Kazakh keeper would achieve a respectable 0.73.

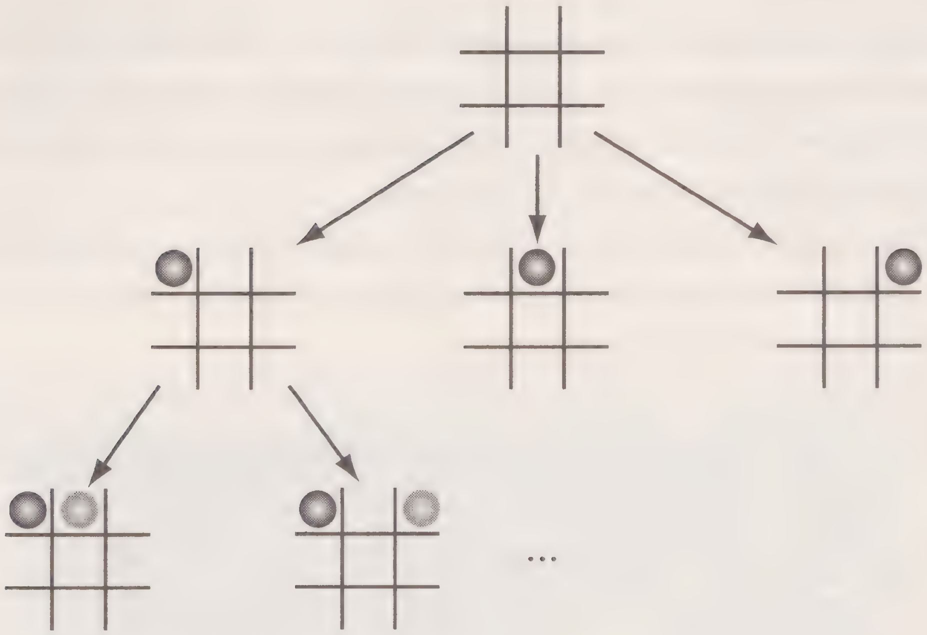
To address this type of problem, conventional search techniques are usually used within artificial intelligence, like back-tracking and branch-and-bound search algorithms. Both techniques operate in similar ways, in essence by deploying a decision tree and combing it until the best alternative is found. In a planning problem, 'deploying a decision tree' just means generating the tree of all the possible planning options (remember the concepts explained in the first chapter when we discussed how an intelligent algorithm can solve a chess problem) and then intelligently 'pruning' those branches of the tree that lead to impossible planning options, violate a certain constraint, or which it can be predicted easily would not lead to a satisfactory conclusion.

The big difference between back-tracking and branch-and-bound algorithms is that the former technique searches the planning tree from top to bottom, while the latter searches it from side to side, and this difference is fundamental, as depending on how the problem is stated, one act of pruning or another may have quite different results in terms of efficiency.

Pruning the tree as the search continues is absolutely necessary since, as in almost any combinatorial problem, if this wasn't the case the number of planning options and, therefore, branches of the tree would become so vast that it would ultimately be impossible to search it in a reasonable amount of time. To shed light on the pruning process, these tree search techniques usually use heuristics tools, which are basically the implementation of a number of intuitive notions that an expert in the field (human or otherwise) can use to determine when a particular branch is not going to lead to a viable solution and prune it as soon as possible. Pruning before or after a non-viable branch can save many minutes or hours of calculations, since the number of options that have to be analysed grows exponentially as the number of levels in a branch grows.

THE 'NO FREE LUNCH' THEOREM

The theorem known as the no free lunch theorem asserts that there is no algorithm capable of solving all the possible problems in the best possible way. The statement of the problem is based on a metaphor on the cost of dishes in different restaurants, hence its strange name. Suppose there are a certain number of restaurants (each of which represents a given prediction algorithm), with a menu linking each dish (each dish being a specific prediction problem) to a price (representing the quality of the solution offered by the algorithm applied to that problem). In this case, we would need a rather greedy, unfussy person who could investigate which restaurant, right now, offered the dish that he most fancied eating at the best price. On the other hand, a vegetarian who accompanied our greedy omnivore would surely find that the vegetarian option was much more expensive or bland. The omnivore, if he fancies a steak, can choose the restaurant that offers it at the best price. But what happens to the vegetarian accompanying him? As it turns out, the only vegetarian dish on the menu in this restaurant is astronomically expensive, but all



A simple example of a planning tree applied to noughts and crosses.

she can do is order it. The long-suffering vegetarian companion very appropriately represents the case in which, given a specific problem, being obliged to use a given algorithm to solve it ensures that the results will definitely be much worse.

The considerations introduced by the no free lunch theorem affect planning, as despite the massive efforts being made in the research community to design a 'superalgorithm' or 'supertechnique' capable of carrying out the best possible planning at any moment, a set of data or a context always eventually pops up in which another algorithm or technique functions better.

Similarly, a corollary of this problem states that if a lot of effort is put into adjusting the performance of an algorithm so that it performs superbly when presented with certain data, this adjustment will harm the algorithm's performance in a different data set; all of this suggests the basic conclusion that either it performs well in a limited set of situations and badly in others, or poorly in all situations.

Conflict detection

The island of Cyprus, eastern Mediterranean, January 1997: The governments of Cyprus and Greece announce the deployment of two batteries of Russian-supplied S-300 ground-to-air missiles, a major reinforcement of the Greek Cypriot armed forces in the context of the Joint Defence Space.

The Russian defence agency publicly asserts that its systems are purely defensive in nature, and will not affect the precarious balance of military power on the island between Turkish and Greek forces.

S-300 missiles displayed in a Russian military parade (source: Archlinux).

Turkey, January 1997: The Turkish government immediately declares this to be a grave threat to their sovereignty over part of the island and begins to take very costly measures to rearm against it. It also declares that if the missiles touch down on Cypriot soil it will attack them, even if it means triggering war on the island.

In the face of these threats, the Greek Cypriot government places its army on maximum alert, and this situation continues until June that year.

Spring 1997: For its part, the Greek government does not see deployment of the S-300 missiles as sufficient to contain the Turkish threat, as the batteries are exposed to a Turkish strike and would not survive such an attack. Therefore, Greece thinks that any attempt to destabilise the region will come from the Turkish side, as the S-300 missiles are purely defensive.

During this period, the Greek army mobilises to support the Greek Cypriot army in the event of a Turkish attack. Russia remains neutral but warns that the sale of the two S-300 systems will go through without any external interference.

Turkey begins rounds of intense diplomacy with its strategic allies in NATO, but to no avail. As a result, it decides to enter into relations with Israel to receive training in operating S-300 systems.

The unstable eastern Mediterranean region.

September 1997: The Turkish navy starts scouring the eastern Mediterranean in search of boats, particularly Russian boats, in order to intercept delivery of the missiles. For this reason, Russia and Greece warn Turkey that they will be at war if Cyprus is attacked or has a naval blockade imposed on it.

December 1997: Russia mobilises major naval forces in the zone, including aircraft carriers, submarines, etc. It is assumed that the purpose of this fleet is to transport the S-300 systems and destroy any Turkish fleet in the event of any attempt to intercept the delivery.

January 1998: In the face of international pressure from the United States and the United Kingdom, and the threat of war from Turkey, Greece eventually decides not to deploy the missiles in Cyprus and, instead, locates them on the Greek island of Crete. Other less powerful batteries and weapons were later deployed on Cyprus, supplied by Greece to the Cypriot government instead of the S-300s. In the wake of this conflict, which could have ended tragically and had serious international consequences, the Turkish and Cypriot governments were destabilised to a major degree. However, this was not true of the Greek government.

But where exactly does artificial intelligence fit into all this? How can it prevent, predict or even advise of the most sensible tactical movements in order to avoid situations that might lead to war like the one we have described? Well, in 2005 a group of researchers from the University of Cyprus revealed a complex intelligent system based on fuzzy cognitive maps trained by scalable algorithms capable of predicting and simulating situations of political instability with great precision. This system, if adapted to the Cypriot conflict, encompasses 16 variables ranging from 'instability/intensity in Cyprus' to 'international influences' via variables like 'Greek political support' and 'reinforcement of the Turkish army'.

A fuzzy cognitive map is merely a neural map in which each neuron computes the strength of a variable over time. For example, at a given moment, Greek political support could be quite strong and the neuron that assesses this variable may be at 92%. Meanwhile the reinforcement of the Turkish army may be weak at a particular time, with the corresponding neuron at 23%.

Each neuron is connected to its neighbours via an edge that considers the cause-and-effect relationship between the two connected neurons. For example, 'political instability in Cyprus' has a 0.32 impact on 'reinforcement of the Turkish army', so that if political instability at a given moment is 50%, this would trigger a direct 16% increase in reinforcement of the army: 0.32 x 50%. There are also negative causal relations. For example, the variable 'solution to the Cypriot problem' has a -0.21 impact on the variable 'political instability in Cyprus'.

The complex causal relations between neurons (in total, there are 45 connections in this cognitive map) are set by a scalable algorithm in which each member of the population represents a matrix of weights evaluated by the 45 connections between the conceptual variables in the map. The fitness of each matrix is measured according to how capable it is of describing past situations of escalating tensions.

So, when the researchers had trained the system with the right weights matrix, they could make simulations along the lines of "what would happen if...", in order to find the best solution to the Cypriot problem. Lastly, an article published by the same authors analysed three scenarios to find out what predictions the system made. In one of the cases they asked the system what would happen if the Turkish army left the island for good; it predicted a whole series of situations of growing instability which ultimately resulted in a situation of chaos and anarchy.

ARTIFICIAL INTELLIGENCE AND WAR

Throughout the course of history, wars have always been a key factor in technological progress, from the development of engineering by Archimedes during the Punic Wars, to the development of nuclear energy in the Cold War. Likewise, artificial intelligence also has its origins in belligerent times. The very fundamentals of artificial intelligence and computing were forged in the heat of the Second World War, driven by the need to decipher the Nazis' secret codes. And during the Cold War the big breakthroughs in this field were born of the necessity to translate large volumes of technical and scientific text from Russian to English. Natural language processing was developed so that these translations could be performed automatically. However, in 1966 the ALPAC (*Automatic Language Processing Advisory Committee*) report was published, in which a governmental committee firmly discouraged continued state investment in natural language processing because of the poor results obtained during ten years of intense research.

Although this story might more properly be classified as a rumour because of its dubious credibility, it is said that during the Gulf War the American army systematically bombed every target that an intelligent system predicted was concealing aircraft. The decisions were seemingly taken automatically using tools, which we will discuss in more detail later, known as 'Kohonen neural networks'.

Enigma machine, used during World War Two to encrypt and decipher messages.

Chapter 5

Data Analysis

The story goes that several years ago a large American retailer, Osco, tasked its IT department with designing a system capable of analysing the enormous volumes of data it created every day, in order to draw conclusions or analyse trends in market behaviour.

Once the system had been set up, one of the first and most surprising trends they discovered was that between 5 and 7 pm the combined sales of nappies and beer increased significantly. In other words, a significantly large number of customers who bought nappies during this time also put beer in their shopping trolleys. This initially disconcerting trend can be explained by the observation that customers with young children can't leave the house in the evenings and so watch baseball, basketball or football games on the TV while they look after the kids, and so they buy beer to drink while enjoying the game.

But how did Osco take advantage of this discovery? Well, when this trend was picked up they quickly moved the beer and nappy aisles closer together, and the combined sales of these products shot up. The example spread and nowadays all self-respecting retail chains use data warehouse-style data mining tools to analyse trends and launch offers on their products.

We will examine the ins and outs of 'data mining' and data warehouses next, but suffice it to say that data analysis is the statistical specialism capable of generating information from systematically collated data. Nonetheless, owing to the growing complexity of the data we get from our environment, it is becoming increasingly difficult to carry out this analysis, to the point that it is now regarded as a discipline halfway between statistics and artificial intelligence.

'Data mining' is the name given to the extraction of knowledge from the information generated by data. Although data analysis goes back to the 17th century, when the first modern states emerged and acquired the ability to gather information systematically on their societies and economies, data mining is a child of the late 20th century, when computer power and new artificial intelligence tools could be harnessed to generate information and thereby extract knowledge from immense quantities of data.

Data mining

A typical data mining process results in a mathematical model that helps to explain the information and trends observed in data, but can also predict the emergence of new trends or even classify or organise data using carefully identified patterns of behaviour.

Data mining takes data, processes it to generate information and extracts knowledge.

The first, and therefore most basic, data analysis tools were based on the conditional probability concepts proposed in the 17th century by the Reverend Thomas Bayes. The problem that really bedevils data analysis projects lies in the very source of the data. For example, imagine we want to analyse data on cancer patients and collate the information from the archives of a hospital specialising in oncology. We would expect to have much more information available on ill patients than healthy patients, as the source of our information is a place that caters for patients suffering from cancer and not those that aren't. This initial deviation is what Bayes set out when introducing the concept of conditional probability, presented in the previous chapter of this book. Bayes' studies of conditional probability gave rise to a set of tools that take account of this deviation in order to compensate for it and produce unbiased conclusions. Generally speaking, a data mining process comprises the following stages:

1. Selection of the data set. At this stage we select the variables we want to segment, classify or predict (also called 'objective variables') and the independent variables, which are the data used to construct models. It is often impossible to work with all the data we have available, so at this stage we also need to select the data we will be working with next.
2. Data properties analysis. At this stage we perform a first, simple study of the data with the aim of identifying atypical or marginal values that lie outside the range of reasonable values. We also reject those variables that don't provide information that can make a significant contribution to solving the problem in question.
3. Processing of input data. At this stage data is usually normalised, because working with non-normalised data tends to give rise to significant errors in the subsequent modelling stages. For example, if two of the variables in a problem are the height and weight of people from a given country, the former will probably be measured in millimetres and the latter in kilograms. If a neural network is then used to model these data, such significant differences in the magnitudes of the input values (a person's height is usually measured in thousands of millimetres, but a person usually weighs tens of kilograms) would cause the modelling tools to malfunction. Data are therefore usually normalised between 0 and 1.
4. Modelling. This is the core phase in data mining. In fact, data mining techniques are classified according to the technique or methodology used in this stage. Modelling methods encompass a large number of techniques and methodologies, like neural networks, SVMs, etc., that are usually derived from soft computing (problem-solving computing techniques that deal with incomplete or imprecise data). These are always designed to extract 'significant information' or knowledge.
5. Knowledge extraction. The tool used at this stage often does not extract knowledge immediately; various tools are implemented at this stage to extract the new knowledge generated from, for example, a properly trained neural network.

6. Data interpretation and evaluation. Despite the intensive use of computerised tools in data mining, this area of engineering is still far from being a fully automatable industrial process. In fact, the consensus is that it is still very much a craft dependent on the experience of the engineer that implements it. For this reason, once the knowledge extraction process is complete, it still remains to be confirmed that the conclusions obtained are correct and do not reveal trivia (for example, that all human beings are between 1.4 and 2.4 metres in height) or falsehoods. Similarly, in a real case of data mining, various methods are used to reanalyse the same data. This stage compares the results obtained using the alternative knowledge analysis and extraction tools.

IS THE POPE AN *ALIEN*?

In 1996, in the prestigious medical journal *Nature*, Hans-Peter Beck-Bornholdt and Hans-Hermann Dubben asked whether the Pope was a human being. They reasoned that if they were to pick a person at random from all human beings, the probability that it would be the Pope would be one in six billion. And taking the syllogistic reasoning analogy one stage further, the Pope has a one in six billion chance of being a human being.

The answer to this fallacy was given by Sean R. Eddy and David J.C. MacKay in the same journal using Bayesian probability. The response stated that the probability that an individual is the Pope, given that he is a human being, is not necessarily the same as the probability that an individual is a human being given that he is the Pope. Using mathematical notation:

$$P(human \mid Pope) \neq P(Pope \mid human).$$

If we want to know the value of $P(human \mid Pope)$, we need to apply Bayes' theorem, which gives us the following expression:

$$P(human \mid Pope) = \frac{P(Pope \mid human) \cdot P(human)}{P(Pope \mid human) \cdot P(human) + P(Pope \mid alien) \cdot P(alien)}.$$

If we assume that the probability that an individual (from planet Earth, of course) is an extraterrestrial is negligible ($P(alien) \approx 0$), the probability that this individual is human tends to 1 ($P(human) \approx 1$), and the probability that an *alien* will be selected to be the Holy Father is rather low ($P(Pope \mid alien) < 0.001$). Therefore, in all likelihood, the Pope is a human being ($P(human \mid Pope \approx 1$).

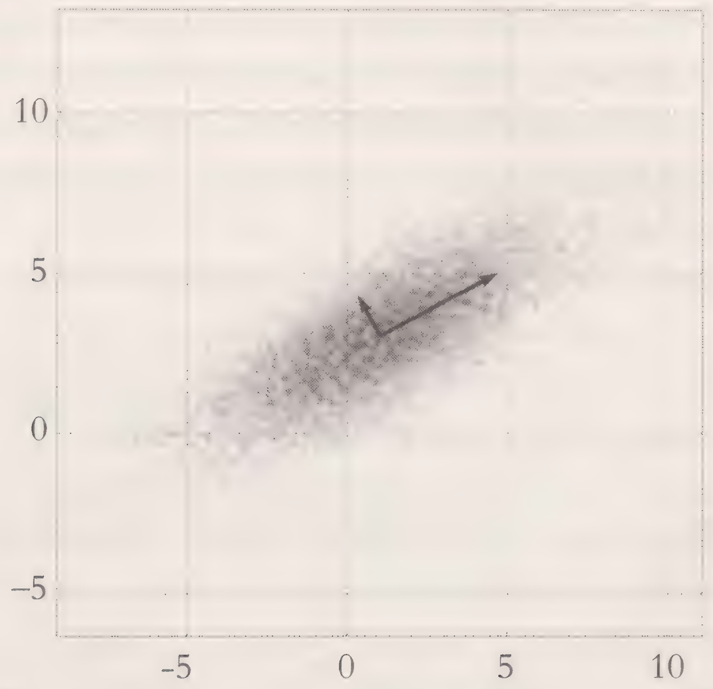
The curse of dimensionality

It is well known that intuition based on insufficient thought is no friend of statistics and probability. There is always a tendency to believe that with a data analysis problem, the more input data (and samples) we have, the more information there is and, therefore, the more knowledge we can extract. However, nothing could be further from the truth. So much so that this fallacy, a common trap that novice 'miners' often fall into, has even been dubbed 'the curse of dimensionality' by experts, and is also known as 'the Hughes effect'.

The problem arises when the volume of a mathematical space is exponentially increased and extra dimensions are added. For example, 100 (10^2) evenly-spaced sample points suffice to sample a unit interval with no more than 0.01 distance between points. However, an equivalent sampling of a unit cube would require 1,000,000 points, or 10^6, and in a 10-dimensional unit hypercube, 10^{20}. Therefore, when adding extra dimensions, in order to maintain equivalence between the proportion of data gathered and the number of dimensions (or in other words, the density of the mathematical space in which we will be working), the data that need to be analysed must increase exponentially. Let's look at an example in practice. Say we want to search for patterns in a country's parliamentary election results and we have a lot of data on voters and their voting preferences. It could be that some of these data, such as height, say, are not relevant to the voting decision. In this case, it is best to eliminate the 'height' variable to increase the density in the mathematical data we will be working with – the collated samples of voters.

As a direct result of the curse of dimensionality, a branch of statistics has grown up called 'feature selection' (also known as variable selection*)*, which uses and combines various mathematical tools in order to eliminate the maximum amount of data that provide no new information on a given problem. This process could range from the deletion of redundant or correlated information to the elimination of random information and constant variables. By constant variable we mean a value that is practically invariable in the data set. An example might be the 'nationality' variable in the analysis of voting tendencies in a single country. It should follow that this variable is the same for all or almost all voters in the country, and will therefore add no value.

The most commonly used feature selection technique is principal component analysis (PCA), which searches for the projection of the data that maximises variance. In the the following diagram, the two arrows represent the two principal components of maximum variability in the cloud of data, in particular the longer arrow. Therefore, if we wanted to reduce the dimensionality of the data, we could replace the two variables plotted on the x and y axes with a new variable, which would be the projection of the data onto the component specified by the longer arrow.



In this graph the arrows indicate the directions in which the data present the greatest variance (source: BenFrantzDale).

PCA aims to find the linear transformation that generates a new coordinate system for the initial data set, such that the first principal component accounts for the greatest possible variance, the second accounts for the second greatest variance, and so on for as many components as required. One of the benefits of using PCA is that in one of the intermediate steps in the search for the components that maximise variance, we can calculate the variability accounted for by each new principal component. For example, the first principal component can account for 75% of the variability; the second, 10%; the third, 1%, etc. We can thus reduce dimensionality while ensuring that the new dimensions that replace the original characteristics account for a minimum level of variability in the data. (It is usually recommended that the variability accounted for by the selected components is around 80%.)

Despite the virtues of PCA and its relative simplicity of execution (nowadays all statistical packages deploy PCA as standard), it has the disadvantage of requiring calculations that grow in complexity as the number of dimensions in the model

WHO'S THIS? FACIAL RECOGNITION

Many modern cameras can detect faces in the frame when the picture is taken. There are now many types of systems and programs that function on the basis of face-detection algorithms. One example is digital cameras that have a function enabling them to detect the number of faces in a photograph and to automatically calibrate the camera's settings to try to ensure that all the faces are in focus. The social networking site Facebook also incorporates a face-detection function capable of suggesting the presence of certain people of interest in the photos uploaded by users. How is all this done?

Most face detection methods operate on the basis of principal component analysis. They rely on training the system using sets of images of different faces, so that the system extracts the principal components from a single person's face and from the set of all faces. In reality, what the system does is to memorise the most characteristic features of each person's face so that it can recognise it in the future. So when given a new image, the system compares the information extracted from this image with the components of its training set by running another principal component analysis. Based on the match percentage, it can detect whether what it is analysing is a face or a shoe, and even recognise which specific person the face in question belongs to.

increases, and the computational cost can get out of hand. In these cases, it is usual to use two feature selection methods, greedy forward selection and greedy backward elimination. Both have two major disadvantages in their large computational cost and the low level of certainty that they will select the most appropriate variables.

However, the fact that they are easy to deploy, the simplicity of the underlying concept and their large number of dimensions, means that the computational cost is lower than PCA. They are therefore popular tools among 'miners'.

As their names suggest, one works forwards and the other backwards, but greedy forward selection and greedy backward elimination are based on the same principle. The best way of explaining both methods is by using a practical example. Imagine we want to select the variables that best explain people's voting habits in a country's parliamentary elections. The five variables in the data set are: purchasing power, city of origin, education, gender and height. And the tool we will be using to analyse trends is a neural network. To begin with, the greedy forward selection method selects the first variable in the problem and, using this variable only, the data are modelled using a neural network. Once the model has been constructed, its predictive power is assessed and the information stored. The process is repeated in exactly the same way with the second variable, and then with the three remaining variables. When the analysis is complete, the variable is selected, the associated model of which delivers the best results and the modelling process using the neural network and model assessment is repeated, but this time with two variables. Let's suppose that the variable that delivers the best results is education. We would then test all the sets of two variables in which education was the first variable. This would give us the 'education and city of origin', 'education and gender' and 'education and height' models. Once again, after the four combinations have been analysed, the best is selected, for example 'education and purchasing power', and the process is repeated with three variables, in which the first two are now fixed. The process continues

GREEDY ALGORITHMS

A greedy algorithm is a type of algorithm with a very specific philosophy. The idea is that in order to decide the next step (whether in a planning, search or learning problem), the option is always selected that maximises, in the short term, a certain gradient, whatever the problem to be solved is. The advantage of greedy algorithms is that they can very quickly maximise a given mathematical function. On the other hand, in complex, multimodal functions (i.e. those with various maxima), they usually end up stagnating at a local maximum, as they can't take an overview of the problem. This is a poor strategy, as the 'optimised' value is usually a sub-optimum.

until, when adding a new variable, the fitness of the model does not improve on the fitness of the model with one fewer variable.

Greedy backward elimination functions exactly in reverse, i.e. it begins with the model that includes all the variables, and proceeds by eliminating those that do not reduce the fitness of the model.

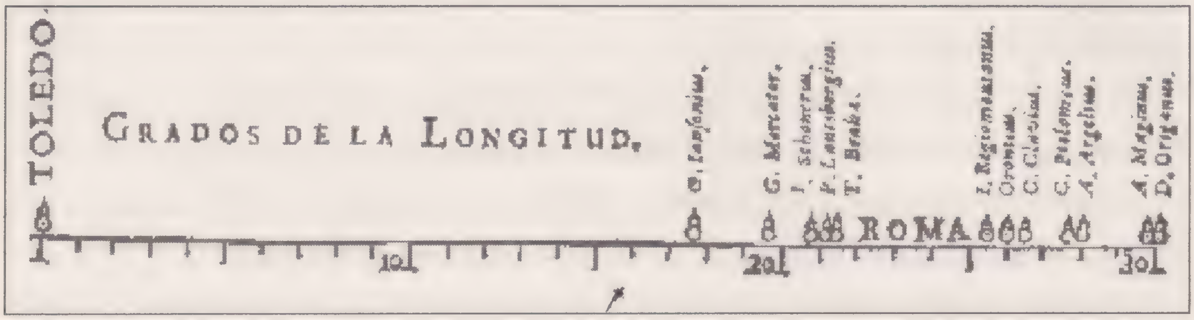
You will have noticed that, despite the simplicity of the method, it remains a 'low-intelligence' model because it does not ensure that the best combination of variables is found, while also imposing a high computational cost because the data have to be modelled at each stage of variable selection or elimination.

The fact that the existing variable selection methods have major drawbacks means that new methods are constantly being proposed in specialist forums. These new methods usually adopt the PCA principle, in that they search for new variables to replace the originals and provide greater information 'density' or 'richness'. These types of variables are known as 'latent variables'. Broadly speaking, they are very commonly used by a large number of disciplines, although they are probably most effectively applied in the social sciences. Descriptors like a society's quality of life, market confidence or a person's spatial awareness are latent variables that cannot be directly observed, but instead are measured and inferred from the measurement of a series of other more tangible variables. Similarly, these latent variables have the advantage that they unite various variables in one, thereby reducing the dimensionality of the model and making it more manageable.

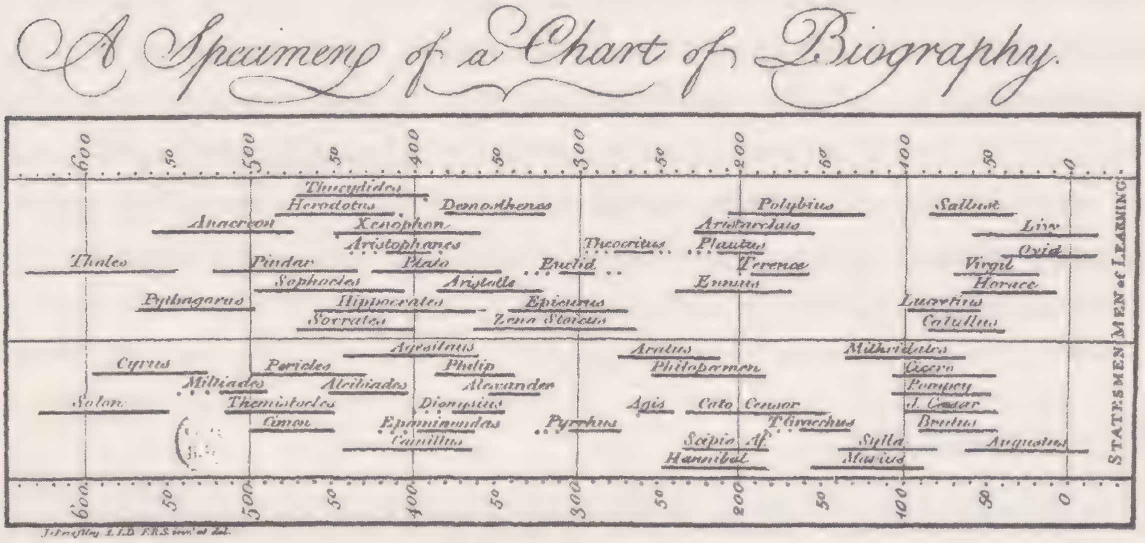
Data visualisation

Data visualisation is the branch of engineering that examines how numerical data, generally multidimensional, can be represented graphically in order to be visualised by a human being. As with data analysis, data visualisation became popular when nascent modern states developed the ability to systematically generate data on changes in their economies, societies and production systems. In fact, this branch of engineering is similar to, and even overlaps, data analysis, in the sense that many of the tools, methodologies and concepts used to facilitate data visualisation spring from data analysis, and vice versa.

It is thought that the earliest recorded visualisation of statistical data was performed by Michael van Langren in 1644, which shows the twelve estimates, made by twelve different scientists, of the distance between Toledo and Rome. The word 'ROMA' indicates Langren's own estimate, and the small blurred arrow in the centre below the line is the actual distance estimated using modern methods.

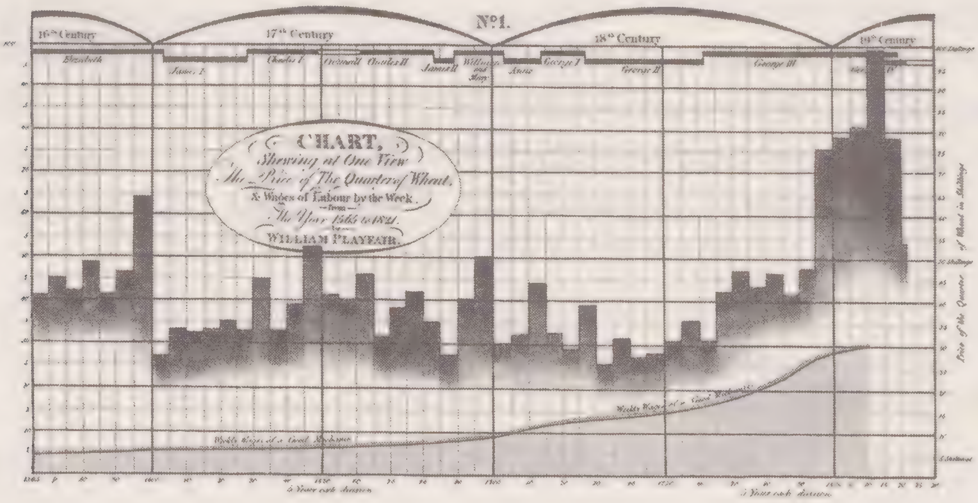


In the 17th century, Joseph Priestley generated the following graphic to show when some of the greatest figures of antiquity lived:



In the same century, thanks to the philosophical reflections of Immanuel Kant, in which he asserted that the existence of the object depends on its representation and not vice versa, people became aware that consciousness or reality could not be discussed without taking into account that said consciousness or reality is constructed by the human mind. This ensured that the great importance of the science of data representation and visualisation was recognised.

Later, during the Industrial Revolution, other, more sophisticated methods of data representation began to appear, such as those introduced by William Playfair to represent industrial and economic output through the changes in wheat prices and salaries spanning various governments and more than 250 years:



From this point, within the field of computer science, data visualisation professionals began to work on understanding what a good data representation needs to look like so that an analyst can interpret it quickly and easily. One of the most important aspects to be taken into account (more so than the technical component of the data, the model of representation and the 'graphic motor' used to visualise them) is the perceptive limitations of the analyst, the end consumer of the data. When this user tries to understand a visual representation of data, he or she executes a number of cognitive processes that build the mental model of the data. But these cognitive processes have major perceptive limitations, such as, for example, the fact that most mortals cannot mentally grasp more than four or five dimensions; these limitations have to be taken into account in order to facilitate construction of the models. Because of all this, a good data visualisation has to show information hierarchically with different levels of detail, be coherent and avoid, as far as possible, any distorted representations. It must also minimise the impact of data that do not contribute information or which may lead to erroneous conclusions, and add other statistics that provide information on the statistical significance of each area of information.

To achieve all this, strategies are used that are similar to those we saw in the section on data analysis. The first is to reduce the number of dimensions, which can be done using the methods we have discussed, like projecting the models onto latent variables. The second is to reduce the number of objects in the model by classifying them into significant groups, a process called clustering. A clustering analysis consists of dividing a set of observations into subsets (also called clusters), so that all the observations grouped in the same *cluster* share certain properties, which may not be obvious.

Clustering the data can make it much easier to represent them graphically in such a way that they can be understood by the human viewer, because of the simplification that clustering brings to the representation. There are many clustering algorithms, each with distinct mathematical properties and thus varying levels of suitability for a given type of data.

Pattern recognition

We couldn't end a chapter on data analysis without talking about pattern recognition, since one of the main objectives of data analysis is to recognise and shed light on any patterns present in order to predict future trends. All the tools discussed so far can be used to recognise patterns: neural networks, support vector machines, principal component analysis, etc. And, as we will see, it is a branch of data analysis strongly linked to artificial learning.

The aim of a classifier system, like a neural network or an SVM, is when given an object, to be able to predict its class, or in a word, classify it. To achieve this, the classifier system must first be provided with a data set of a known class so that the system can learn. Once the system has been trained, new data can be fed into it for classification. As with previous methods, the initial data set of a known class is usually divided into two subsets, the training set and the test set, which are subsequently used to check that the system is not overtrained.

Learning classifier systems come in two types: the Michigan LCS (so named because researchers at that state's university promoted it) and the Pittsburgh LCS (for the same reason). A Michigan LCS is nothing more than an evolutionary algorithm in which the individuals that evolve are rules (classifiers), each rule made up of a set of conditions and a purpose. The idea is that if an object matches the conditions imposed by a rule, the object's class will be that indicated by the rule's purpose.

In Pittsburgh classifier systems, meanwhile, each individual is a set of rules and the individual's fitness is assessed on the basis of the average error rate of each of the rules in the set. Both systems, which have quite a few complementary features, have their advantages and disadvantages. In the last 30 years, researchers from all schools have proposed a series of improvements and variations on both models in order to do away with their inefficiencies.

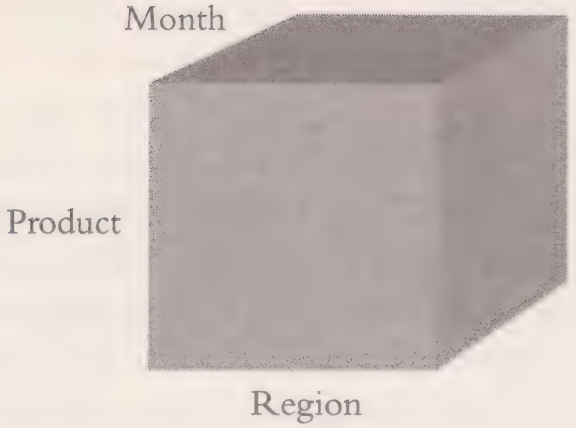
A practical example: sales analysis

Another of the main business applications of artificial intelligence is *data warehousing*. A data warehouse is a business tool widely used by companies with large customer bases and, therefore, a large base of data from which to infer behavioural trends, fashions or patterns. It is where a company's full data archives are kept, whether from sales, production, marketing campaigns, external sources, etc. Data warehouses are now used in organisations as varied as banks, hospitals, food retailers, oil product manufacturers, governmental organisations, etc.

Creating and structuring a data warehouse is a complex task that can take specialist engineers months or even years. However, once they are constructed, structured and consistency-checked, data warehousing technologies use a concept called 'OLAP cubes' (in fact they are hypercubes), which study and analyse the data. An OLAP (OnLine Analytical Processing) cube, is a multidimensional data structure which makes it possible to perform very fast cross-checks between data items of different types. In can be seen as a multidimensional extension of a spreadsheet. For example, if we constructed a table in a spreadsheet where we listed which dairy products were sold in different countries last year, in thousands of units, we might produce a table like the following:

	UK	Italy	France	Germany
Natural yoghurt	4,540	5,312	5,429	10,234
Lemon yoghurt	8,988	14,543	11,234	26,342
Peach yoghurt	12,349	16,234	15,345	23,387
Yoghurt drinks	1,676	2,221	3,234	1,476
Custard	4,678	6,934	4,343	1,893
Rice pudding	5,122	7,300	8,345	345

Next, we would like to be able to break down these data down by month, adding a third dimension to the table, so that for each region and product type we have a breakdown into the 12 months of the year.



Once the cube has been structured, complex data analyses can be carried out taking account of this pre-calculated structure. The main computational cost of a data warehouse is not the data analysis itself (in which many of the tools discussed throughout the chapter are often applied) but rather the construction of so many hypercubes with all the dimensions that the organisation's data can support, taking account of the multiple combinations possible. The generation of OLAP cubes is a process which organisations usually carry out at night, to examine and analyse the previous day's data.

So, the analysts of a company which produces dairy products might add to the system the weather conditions on each day of the year in each region where the company operates. This new dimension would make it possible to carry out studies of trends in the consumption of different products according to the ambient temperature on each day of the year in each region.

Next, using this knowledge and the weather forecasts for another year, the analysts can predict the number of units that need to be produced in each region in order to minimise unnecessary storage of dairy products, which has a high cost because of the cold chain that has to be maintained throughout the life of the product.

To complicate the concept of dimensions in an OLAP cube a little further, hierarchies are often added within a single dimension. In this way, to pursue the previous example, a level lower than a month might be added to the time dimension, perhaps a day, or a higher level, perhaps a quarter. In all probability a different quantity of dairy products will be consumed in winter compared to summer, both at the beginning and end of the month. Another dimension which might be split into levels is the region. A higher level could be added, which might cover the south of Europe, central Europe, etc., or a lower level could be added, such as Lombardy, Brittany, Gwent, etc.

Of course, once the OLAP cubes are constructed, apart from the obvious data analysis, which we have already discussed, many other data visualisation operations can be performed. For example, we might visualise two-dimensional 'slices' of the cube, visualise 'mini cubes', i.e. small multidimensional portions of the cube, add or subtract information using hierarchies, or even rotate the cube so that the data can be viewed from another perspective.

MICROSOFT RESEARCH

The biggest private non-academic artificial intelligence research centre in the world is currently Microsoft Research. It is home to internationally renowned scientists who are researching topics such as artificial learning and new man-machine interactions, both of which are critical to this field. Microsoft Research has branches virtually worldwide, for example in Germany, USA, UK, China, India and Egypt.

One area in which this centre leads global research is the use of Bayesian networks and other probabilistic tools for such important purposes as the detection of unwanted email ('spam') or the intelligent adaptation of operating system interfaces to user behaviour, so that future user interfaces might intelligently adapt to each person's working habits.

Chapter 6

Artificial Life

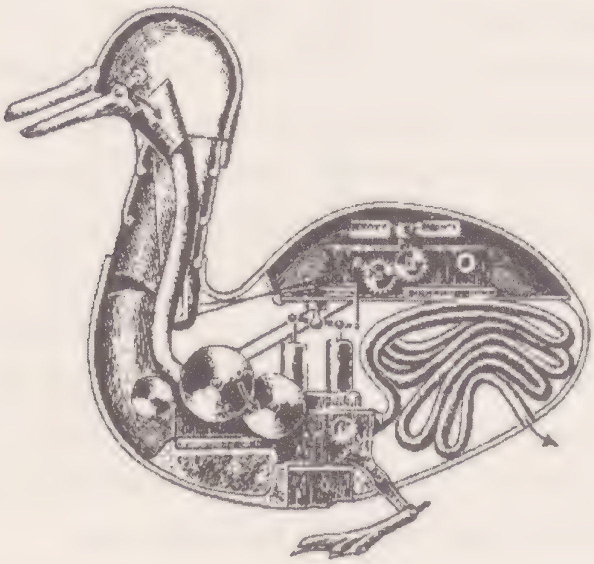
Life and intelligence are two concepts that are equally difficult to define, not only in the field of biology but also philosophically. Searching for formalisms to define life is probably as complex a task as searching for formal definitions of intelligence. Readers will remember from the first chapter all the philosophical and mathematical discussions about delimiting the concept of intelligence – the Turing test, the China room, debates about creativity, etc. One of the most acclaimed and active authors in this field, John H. Holland (b. 1929), who was also the architect of evolutionary algorithms, has delved deeper and deeper into this question over the years, coming to conclusions which will help us to understand this concept.

Artificial life is closely linked to another important concept in artificial intelligence: soft computing. Soft computing is a set of tools, generally inspired by processes present in the natural world, which solve highly complex problems, as the information they deal with is approximate, uncertain and incomplete. Some of these tools are the ones we've been looking at throughout this book, including evolutionary algorithms, neural networks and fuzzy logic.

Soft computing became a formal branch of computing in the 1990s, and is now used to solve problems to which the experts have resigned themselves to not finding the best solution. In some cases finding a good solution to these problems might take years of calculation or require information that is impossible to compile. All areas of modern science and engineering, from biology to political science, now use soft computing to solve problems.

An introduction to artificial life

One of the most important and essential requirements for a system where there is 'life' is that certain conditions are met so that self-organising systems that are much more complex than their individual parts can emerge from the environment. A good example of this is an ant colony, where the relatively simple behaviour of uncomplicated components – the ants – gives rise to a completely self-organising system – the colony – which is of course much more complex than the sum of its parts. Another characteristic of life is that the supposedly 'living' entity must also be able to survive in its environmental conditions and be able to reproduce. Similarly, for something to be said to be living it has to demonstrate a degree of non-random dynamism that is independent of any change in the laws that govern the environment. And a living system has to display emergent, recurrent behaviour, without of course fully adopting regular patterns of behaviour. In order words, if a process has emerged that displays noticeable behaviour patterns, but this behaviour is cyclical or looping, the entity that exhibits it cannot be regarded as 'living'.



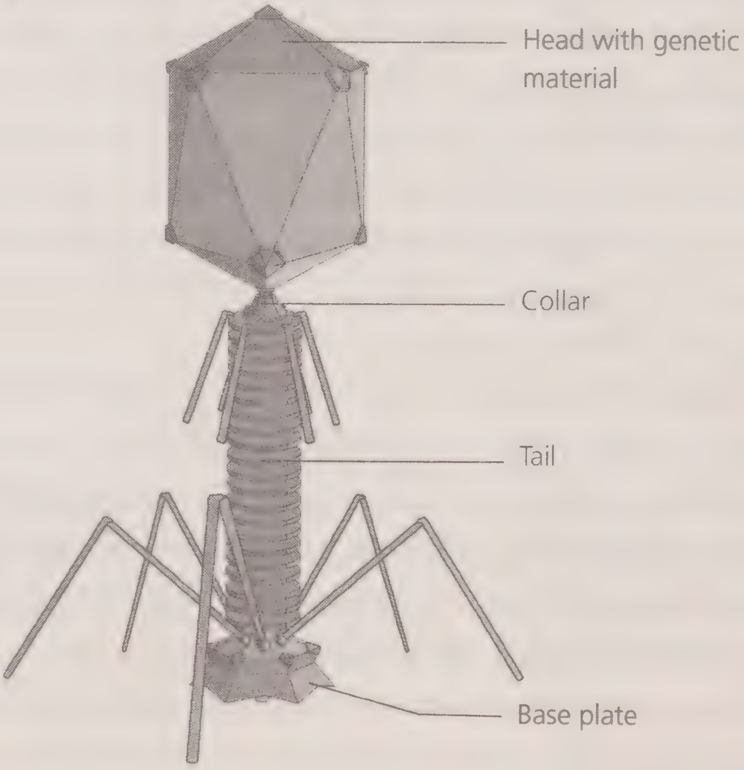
Jacques Vaucason's (1709–1782) duck; Vaucason was probably the first engineer to take an interest in artificial life.

As readers can appreciate it is not easy to define life with any degree of mathematical precision, and there are no simple criteria for determining whether an artificial or natural entity can be regarded as living. This debate, as we noted earlier, goes beyond mathematics and computing. In fact, in biology there is still no full consensus among scientists on a number of questions relating to life, such as whether or not a virus is a living entity. So, in the light of these definitions, is a computer virus a living entity? Computer viruses display dynamic, appreciable behaviour that is neither regular nor cyclical. However, can we state that a computer virus has emerged naturally? Probably not, because ultimately it was created by a malicious computer programmer, unlike a natural virus, which emerges naturally.

VIRUSES AND PRIONS

Viruses are biological systems that are incapable of reproducing alone; as a result, the vast majority of biologists believe that they are not living things, although there is no consensus on this. There are many different types of viruses, but they all have in common the presence of genetic material, which they inject into a host after entering it, thereby ensuring that the host makes copies of the viruses including, of course, the genetic material. These copies spread throughout the organism to infect other host cells.

A prion, meanwhile, is a much simpler entity, with no genetic material, but one which can be propagated between organisms. Prion transmission mechanisms have yet to be explained by the scientific community, but prions are entities of great scientific interest because they cause serious illnesses, such as bovine spongiform encephalopathy, or 'mad cow disease'. Prions are natural proteins in a misfolded form. When these proteins come into contact with an organism, they induce existing, properly folded proteins to convert into the misfolded form, and thus cease to function properly; the new proteins are capable of propagating this misfolding 'infection' to other proteins. Are proteins living things? Biologists think not.



Basic diagram of a virus.

LIVING COMPUTING

This chapter gives examples of artificial systems that imitate the behaviour of living things, but also aims to show their mirror image – computing systems constructed from living things. The processing unit in a modern computer comprises hundreds of millions of transistors, units that use electrical impulses to perform all the operations. Transistors are lifeless objects created using inorganic elements, like silicon. But is it possible to replace transistors, simple metal units, with living systems created using cells? Researchers specialising in biology and physics have recently managed to do precisely this, using living cells to compute the mathematical operations, as transistors do. In the future, therefore, we might see artificial life systems supported by a biological computing platform. Will the computers of tomorrow be living things we have to supply with food rather than electricity?

Complex adaptive systems

To simplify the definition of the concept of life, some experts have developed another more general concept, that of the 'complex adaptive system'. A complex adaptive system is an agent or set of agents that work in a coordinated manner and is or are intelligent enough to be able to adapt to the environment, depending on the behaviour that other systems display. In fact, the definition of a complex adaptive system encompasses living systems that go far beyond what may come to mind when we think of a 'living being', such as a human being's immune system, a commercial company or a whole ecosystem.

The case of the company is quite strange, as surely no-one would have said that a legal entity like a company could have 'life'. But if we think about it carefully, a company is an entity that is born, grows, can reproduce and may die. In most countries, a company has almost the same rights and responsibilities as a person, to the extent that both share a definition. Some are natural persons (human beings) while others are legal persons (companies).

There is a degree of consensus on which seven characteristics define a complex adaptive system – four properties and three mechanisms. In addition, the combination of these basic characteristics gives rise to other composite properties and mechanisms. The definition of a complex adaptive system is slightly more general than the narrow biological definition of 'life': we would never say that a financial entity or city is a 'living being', for example. Therefore, the fuzzy and hard-to-define term 'artificial life' is not widely used, outside of sensationalist news stories and non-specialist forums.

First property: aggregation

Aggregation involves summing the behaviour of simple entities, such that from it emerges an aggregate behaviour that is rather more complex than the sum of their individual parts. (Think of the example of the ant colony and its component ants, in which the adaptability of the colony as a whole to environmental changes is much greater than the adaptability of the individual ants.) Each one of these individual parts is called an agent.

In addition, this property is recursive, and an agent that emerges as the aggregation of other simpler agents can aggregate again with others, either of its own or a different species, to form another second-level aggregated agent. For example, the aggregation of the behaviour and productivity of all companies in a country, plus family consumer behaviour and government spending, equals a country's gross domestic product.

We should stress that a factor that is not covered under aggregation but is essential for the emergence of environmentally adaptable behaviours is communication between members of a category (the first meaning of aggregation) or between the distinct components of the higher-level entity (second meaning).

First mechanism: tagging

Tagging is a mechanism that actively facilitates agent aggregation. Agent tagging is as simple a concept as the posting of marks or signs, and not only do they make agents easier to identify, they also help to break the symmetries that can often form in the aggregation of complex systems. For example, if a white snooker ball starts to roll in one direction and its surface is absolutely mark-free, onlookers will find it hard to see that the ball is rotating, still less work out the speed of rotation. On the other hand, if a sign is printed somewhere on the surface, except at the two points where the axis of rotation intersects the surface of the ball, the audience will be able to determine the direction and magnitude of rotation.

SIMCITY AND COMPLEX ADAPTIVE SYSTEMS

Cities are good examples of complex adaptive systems, as they present and implement the properties that define these systems and more. In fact, the computer game *SimCity*, in which gamers can build and simulate cities, is a fantastic test environment for familiarising people with a complex adaptive system, as this game 'fills' the city with citizens, generates social and commercial activity within it and presents gamers with complex situations to be resolved, such as jammed communications links and natural disasters.

Another computer game we can use to familiarise ourselves with complex adaptive systems is *Civilization*, in which the aim is to build an entire competitive civilisation, with cities, communications networks, trade treaties, defensive systems, social and scientific policies, etc.

An image of a city built using SimCity.

A great variety of signs are used by aggregated agents, from the eagle standards flown by Roman legionaries to identify each legion, to the complex tags with which modern telecommunications systems mark the frame of sent messages. Not only is the structure of each frame indicated by the tag so that the message can be reconstructed once they all arrive at the receiver, but there may also be sophisticated mechanisms to identify any errors that may have affected the message or the tag itself during the transmission process. Of course, not all tags have to be visible. For example, mammals of different genders belonging to some species tag themselves during rutting periods using chemical messengers called pheromones.

For agents, the tags facilitate selective interaction, on the basis of which the agents can distinguish between various instances of the same class of agents and the various aggregated parts of an agent. This can give rise to the use of filters or cooperative plans. The agents can also remain aggregated; although the various aggregated parts that make up the higher-level agent continue to change, the tagging remains. In short, tagging is a mechanism that facilitates organisation and communication between agents.

Second property: non-linearity

It is not widely known that most of the tools that mathematics has given us are linear in nature. From arithmetic to algebraic topology, via differential calculus, they are all based on assumptions of linearity. A function is linear if its value, for any value assigned to its arguments, is merely the weighted total of the sum of these values. For example, the function $4x+2y-z$ is linear; conversely, $4\sin x-2y^{-z}$ is not linear.

The use of linear instruments is so important in mathematics and engineering that a large part of a modern engineer or scientist's professional activity consists in searching for linear functions that approximate natural phenomena as accurately as possible. Unfortunately, however, none of these tools functions effectively in complex adaptive systems. In fact, one of the characteristics that best defines these systems is the fact that their overall behaviour is substantially more complex than the sum of their individual parts, which by definition gives rise to non-linearity.

A good example that clearly illustrates the non-linearity of nature and complex adaptive systems is producer-consumer dynamics and, in particular, prey–predator dynamics. Imagine a wood where there are D predators (foxes, for example) and P prey (in this case, rabbits). If the probability that a fox catches a rabbit is c, then every day $c \times P \times D$ rabbits will be hunted. For example, if $c=0.5$, $D=3$ and $P=10$, then $c \times P \times D = 0.5 \times 3 \times 10 = 15$ rabbits will be caught. However, if the number of foxes and rabbits increases fourfold, the number of captures does not: $c \times P \times D = 0.5 \times 12 \times 40 = 240$. As you can see, predatory activity cannot be calculated simply by adding the new predators to the prey. Even in a relatively simple situation, non-linearity can have a huge effect on an aggregated system. That is why it is always said that the aggregate behaviour of a complex adaptive system is more complicated than the individual behaviour of its component parts.

Third property: flow formation

Flows appear at all levels within complex adaptive systems, where there must always be nodes, transporters and the transported resource. Examples of complex adaptive systems with flows include a human's central nervous system, where the nodes are neurons, the transporters are the synaptic connections between them and the transported resources are electrical impulses. Another example is flows within an

LOTKA–VOLTERRA EQUATIONS

The equations in the foxes and rabbits scenario could get quite a bit more complicated. In fact, a researcher called Alfred J. Lotka discovered what would happen to these equations if we began to take into account variations in the numbers of predators and prey over time.

Suppose that $D(t)$ and $P(t)$ are, respectively, the number of predators and prey at a moment in time t. We could also say that at any moment in time n predators could be born and m predators die. Therefore, the formula for the changing numbers of predators and prey over time is $D(t+1) = D(t) + nD(t) - mD(t)$, and the same would happen for prey: $P(t+1) = P(t) + n'P(t) - m'P(t)$. We have to take into account that, in the case of predators, a greater number of prey means more births; this is expressed, for example, using the constant r, which describes how efficiently food is transformed into descendants. The number of prey–predator encounters, as we saw earlier, is cPD; thus the new equation for predators would be:

$$D(t+1) = D(t) + nD(t) - mD(t) + r[cP(t)D(t)].$$

However, the reverse happens with prey, as every time there is a prey-predator encounter, the number of the former decreases; therefore, the equation would be:

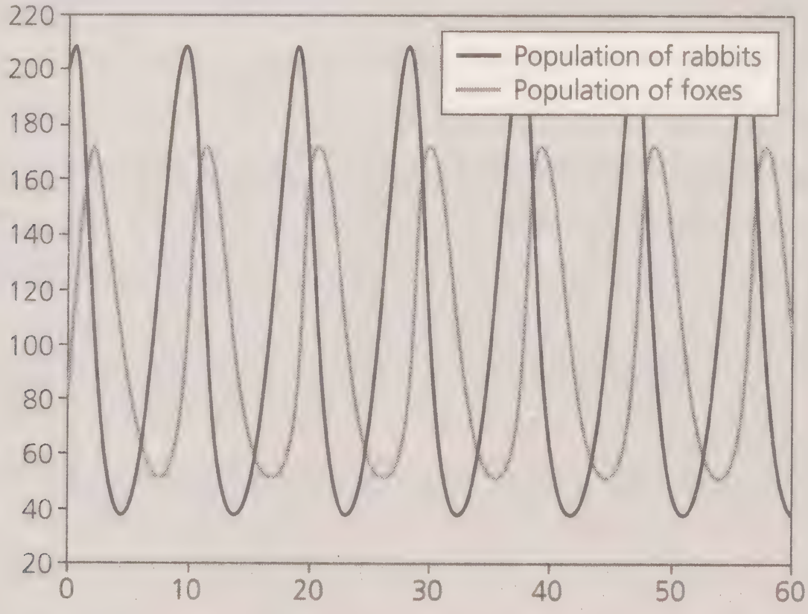
$$P(t+1) = P(t) + n'P(t) - m'P(t) - r[cP(t)D(t)].$$

ecosystem, where the nodes are species, the transporter is the food chain and the transported resource is the energy represented by biochemical elements (proteins consumed, sugars, etc.).

Generally speaking, nodes are processors of the resource, and the connections define the interactions between them. However, in a complex adaptive system we have to bear in mind that the network of interactions can be changeable, and the nodes and connections can appear and disappear. It is this that makes a complex adaptive system an entity that can adapt to the environment and which can, independently, continue to modify its behaviour according to the needs of the moment, suitable or otherwise.

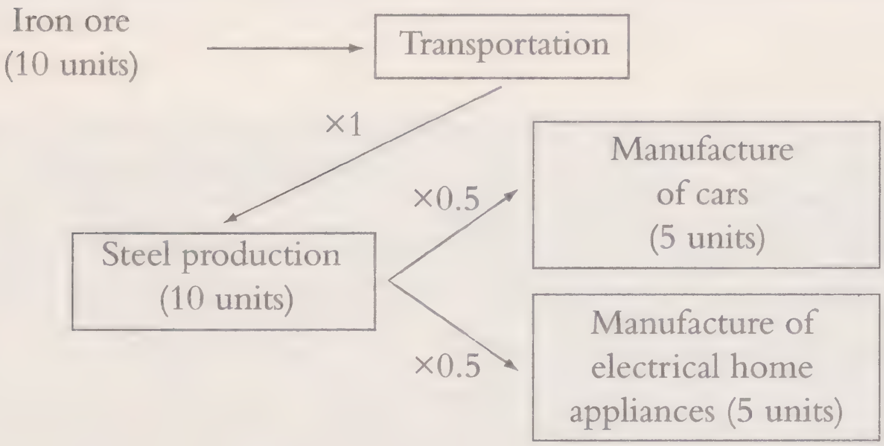
Tagging is one of the most important mechanisms of complex adaptive systems for defining flows. In fact, tags can identify the critical connections for the

If we now consider these two equations, set the constants and resolve them from one point in time to another, we will see that $D(t)$ and $P(t)$ oscillate over time, and prey and predators go through continuous cycles of feast and famine.



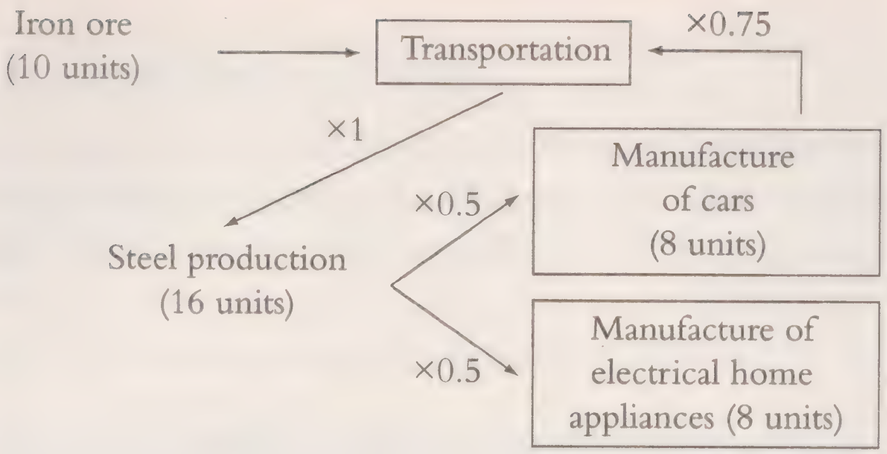
Graph representing the changing populations of rabbits and foxes over time, according to the Lotka–Volterra equations.

transportation of resources. Flows have two properties that are interesting in terms of the functioning of complex adaptive systems. The first of these is the multiplier effect they bring to the system. For example, in a complex adaptive system, like a country's economy, the effect of transporting money from one node to another (like between banks) acts as a risk multiplier. The second interesting property is the capacity for creating cycles, which is why cases of recycling may occur. For example, you can see from the following diagram how industrial production increases in non-linear fashion in a complex adaptive system like a car production line, with and without recycling.


Iron ore
(10 units)
Transportation
×1
Steel production
(10 units)
×0.5
Manufacture
of cars
(5 units)
×0.5
Manufacture of
electrical home
appliances (5 units)

In the initial scenario, the steel producer processes, with a 100% efficiency rate (or × 1), the iron ore into steel. Next, 50% (or × 0.5) of the steel produced is used to make cars, and 50% to make domestic appliances. If, to keep things simple, we suppose that for each unit of steel one car or one domestic appliance is produced, at the end of the flow we will have produced 5 car units or 5 domestic appliance units.

Now let's see what happens with recycling as a multiplier agent:


Iron ore
(10 units)
Transportation
×0.75
×1
Manufacture of cars (8 units)
×0.5
Steel production (16 units)
×0.5
Manufacture of electrical home appliances (8 units)

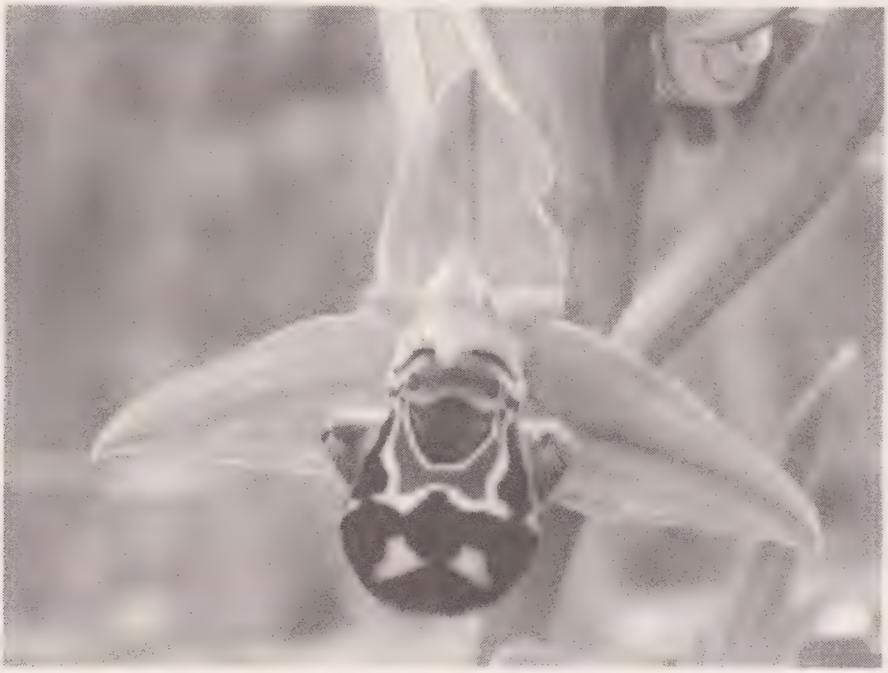
In this second scenario, 75% of the cars are recycled; therefore, the steel producer is now capable of producing more steel, which is ultimately converted into more units of produced cars. If the system begins with 5 units of recycled cars, productivity will increase from one cycle to the next, until the system stabilises at 8 units of produced cars and, therefore, 6 recycled units. This means that steel production increases to 16 units: 10 of which come from the 10 units of iron ore and 6 are produced using recycled cars.

Fourth property: diversity

Diversity is another of the characteristics that define complex adaptive systems. In any adaptive system there is a great diversity of agents that, once coordinated, form the system's patterns of behaviour. By way of example, in a rainforest it is possible to walk for half a kilometre without coming across the same species of tree twice. But the rainforest is not the only system that is home to such diversity. Now consider another complex adaptive system, a city like Rome, with millions of different people, each one with his or her own trade and specialisms, and thousands of shops and businesses, each one, for the most part, completely different from the others; each of these shops and businesses is in turn a complex adaptive system.

This diversity is neither accidental nor random. Each agent within a system occupies its own behavioural niche, which is defined by the connections that have been established with neighbouring agents. If an agent is removed from a complex adaptive system, the system adapts so that other agents automatically fill the 'hole' it leaves. When the system has stopped adapting and arrived at a stable situation, it is said to have converged.

Diversity also appears when an agent or set of agents expands into new behavioural niches, providing the opportunity to create new functionalities that can be exploited by the complex adaptive system. A good example of this is the process of mimicry. As an example, an orchid evolves so that its flowers imitate the shape of an insect in order to trick real insects into landing on them and thereby transfer pollen more efficiently between blooms.



Ophrys apifera, *the bee orchid, grows flowers that imitate the shape of insects to attract them (source: Hans Hillewaert).*

But the key question that researchers usually ask is: "What is it that enables and even motivates a complex adaptive system to create such diversity?" Well, detailed studies of a system like this usually reveal, step-by-step, which adaptations they have undergone so that one agent or another emerges, shedding more light on the need for each of them. For example, in a scenario where the complex adaptive system has adapted to create cyclical flows and, therefore, recycle and increase efficiency generally, niches open up which give rise to the emergence of new agents, such as 'recycling agents'. Another scenario which gives rise to diversity is a growing company. This is a system that requires the appearance of new hierarchies, and in turn the presence of another type of agent which coordinates each hierarchical level.

Second mechanism: internal models

Each adaptive system has a certain ability to create its own internal model of its surrounding environment, which provides, above all, a vision of future events and the changes that need to be made so that it can successfully adapt to them. These models of the environment are constructed on the basis of information flows received by the system and, subsequently, these flows are transformed into useful internal changes that shape the models.

Once the model is constructed, this helps the system to anticipate the consequences when a given pattern appears in the environment. But how can a system transform experience into models? How can a model be developed to be able to anticipate the consequences of future events?

As always in the natural world, evolutionary pressure is the best tool for constructing this type of mechanism. The fact that a bacterium 'knows' that it always has to follow the direction marked by the maximum food gradient is an 'instinct' determined by an internal model that informs it that by following this pattern of behaviour it will maximise its chances of securing the source of food. If a bacterium has not managed to create the internal model that gives it this instinct, it has less chance of reproducing and, therefore, leaving descendants. Bacteria that have codified the structures and hierarchies between its internal agents that give it the capacity to create this internal model will have more chance of reproducing and, therefore, extending this property to the rest of the population.

There are two different types of internal models: tacit and overt models. The example of the bacterium that follows its instinct in search of more food is a tacit model, as it is a model that does not enable it to 'think' or simulate what would happen if it did something else.

Conversely, an overt model, which appears in nature in higher-level entities, is a tool that does give its possessor insight into various hypothetical scenarios, enabling it to take the best decision after analysing the various alternatives. An example of internal modelling in a computerised, complex adaptive system is a chess-playing machine, capable of analysing hundreds of thousands of moves at each turn before moving a piece. Naturally, when the model is tacit, it is created and adapted to the environment at an evolutionary scale; if it is overt, the speed of adaptation is much greater.

Set of bacteria of the species Escherichia coli *magnified 10,000 times. Each 'rod' corresponds to one bacterium.*

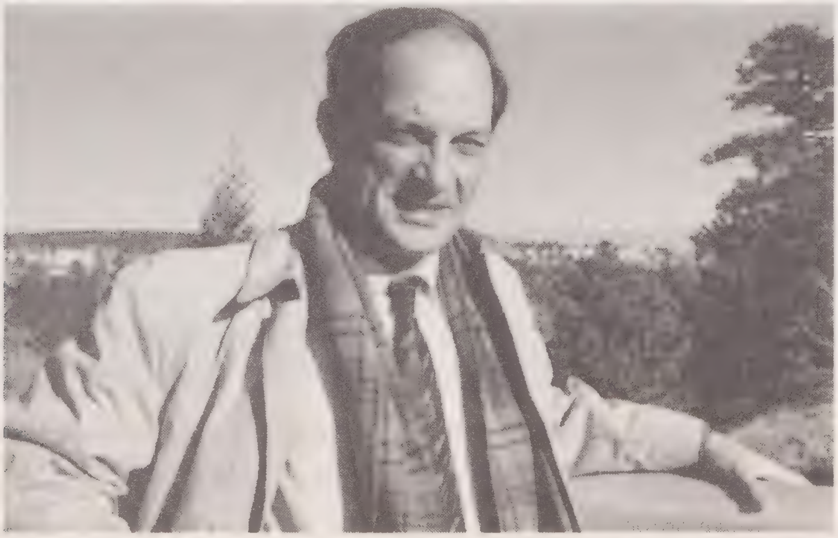
Third mechanism: building blocks

A complex adaptive system's internal model is normally based on a set of limited samples of past situations that are in some ways similar, although each one of them builds in a new aspect compared to the previous one. But how can a complex adaptive system create internal models based on limited past experiences that are useful in new future situations? The key to this paradox lies in the use of so-called building blocks. These are the elements into which any system, set or scenario can be broken down. For example, consider the case of a financial entity that, in a sense, fully meets the definition of a complex adaptive system. Suppose the entity is trying to decide whether or not to grant a loan to a new customer, and its main concern is to determine whether the customer in question is capable of paying it back in the agreed time. The bank has no idea whether the customer will be able to continue to make the repayments in 15 years, as it cannot predict the future. Suppose, to make things more difficult, that the customer is also completely new, with no credit history, and therefore no previous reference. What the bank would do in this case is to break the problem down and by analysing the defining characteristics of the new customer, such as level of education, profession, marital status, etc., see how customers with the same profile have behaved before. For a bank analysing a new customer, these characteristics are the building blocks that define the scenario facing this complex adaptive system.

The capacity to combine building blocks to produce tacit internal models is formed at an evolutionary scale, while learning in overt internal models is a capacity that is usually formed at much more reduced scales, although in the natural world it only manifests itself in higher-level animals.

Cellular automata

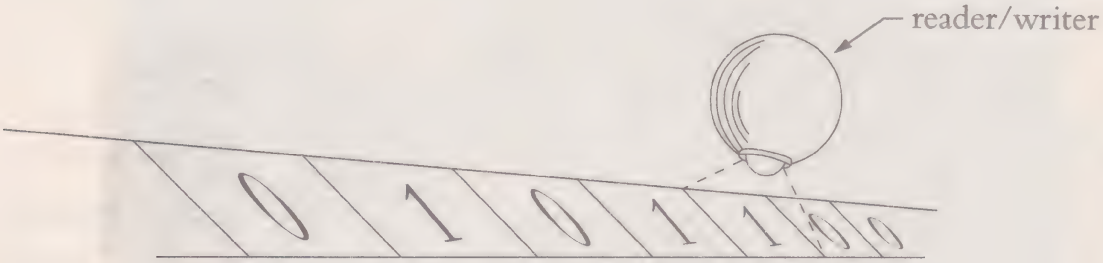
The most classic example of artificial life (or rather, complex adaptive systems) in the field of computing is that of cellular automata. A cellular automaton is a fairly simple concept that helps to explore the complexity of higher systems, and was the fruit of research by two of the most renowned computing researchers, Stanislaw Ulam (1909–1984) and John von Neumann (1903–1957), who were also great friends.



The Polish–American mathematician Stanislaw Ulam.

Generally speaking, automata are mathematical formalisms which, faced with a given input, execute a series of previously programmed instructions. In other words, an automaton is a generalisation of an algorithm or a computer program. In computing, automata are everywhere, from microchips programmed to perform certain actions to operating systems. We saw an example of an automaton in the first chapter – the Turing machine.

Theoretical automata, like the Turing machine, are usually instruments that read their inputs and print their outputs on one-dimensional tape. The automaton thus travels over the tape from left to right, reading the symbols written on it, as the following diagram shows; based on these symbols and its programming, it performs one action or another, including printing a given symbol on one part of the tape.

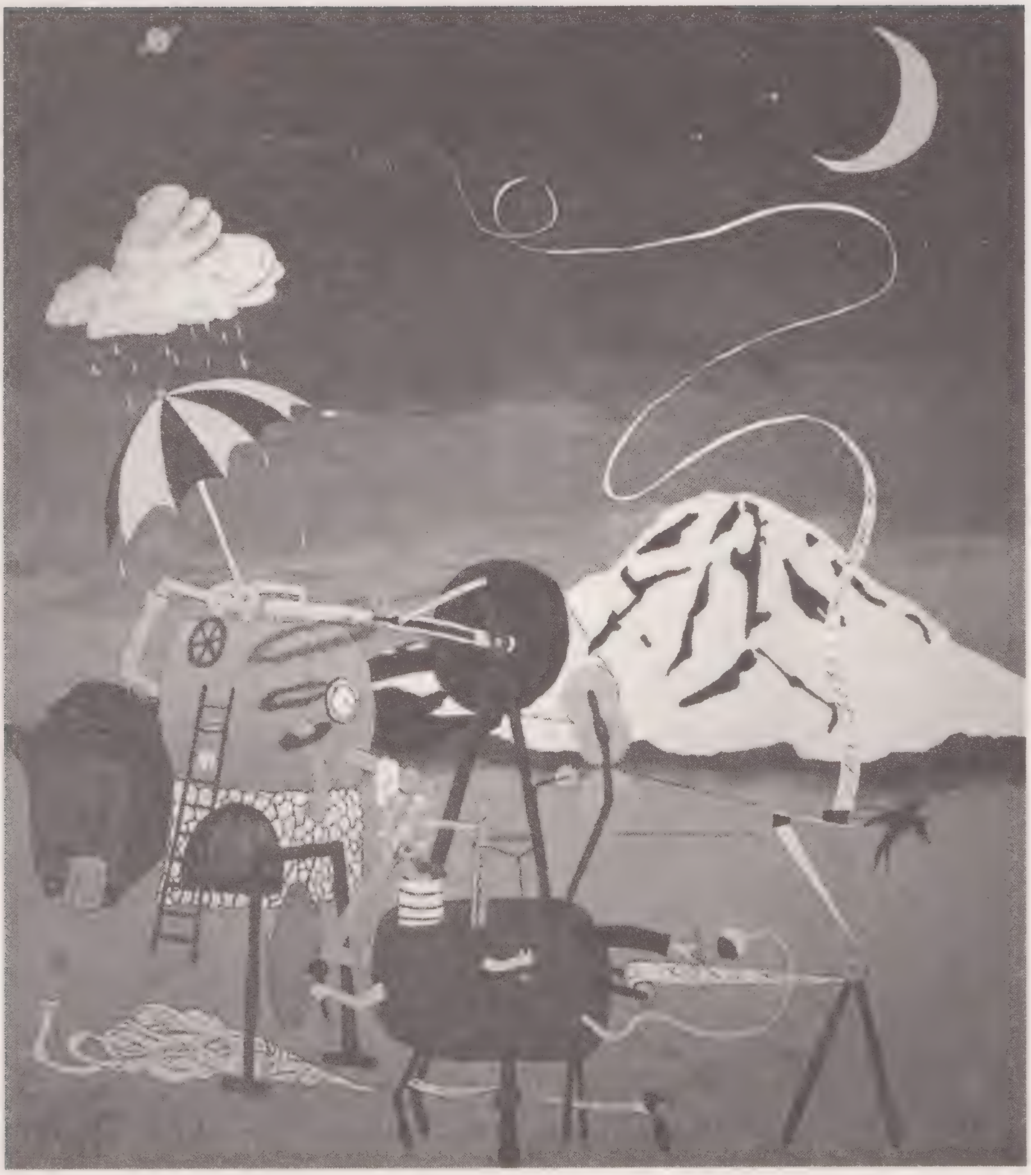


Two of the basic components of a Turing machine. The strip of tape and a reader able to write (source: Complexity, *by Melanie Mitchell).*

However, cellular automata belong to a particular class of automata that do not travel over two-dimensional tapes; instead the input/output medium is a flat chess board-type grid, and in each of the squares (cells) there is a cellular automaton that doesn't move. In cellular automata the information inputs are the cells adjacent to the cell in which the automaton is situated, and the information output is delivered in the cell in which it is located.

Each automaton placed in each of the cells in the grid is programmed with a series of instructions. For example, if there is an even number of black cells surrounding the cell where the cellular automaton is located, it paints the output cell black; otherwise, it paints it white. Accordingly, and by putting a cellular automaton in each cell in the grid, the grid can display various forms or patterns, which change according to what colours the cellular automata are painting the cells at a given moment.

The infinite configurations that a cellular automaton can take include a set which give rise to the emergence of perpetual events, as is the case with Conway's automatism or the Game of Life. An internet search quickly reveals a large number of configurations that generate pretty patterns that are created, self-destruct and re-emerge, all programmed using relatively simple rules similar to those used in Conway's automatism.



Picture of a steam-powered Turing machine painted by students at the University of Washington in one of the university's classrooms.

Artificial immune systems

Imitating the wise and intelligent behaviour of the natural world has always been a great source of inspiration for engineers specialising in artificial intelligence. Nature inspired the development of neural networks and evolutionary algorithms, which we have already explored and which are of great importance to the history of artificial intelligence. The same source of inspiration gave rise to other techniques, like artificial immune systems, which attempt to imitate the behaviour of an

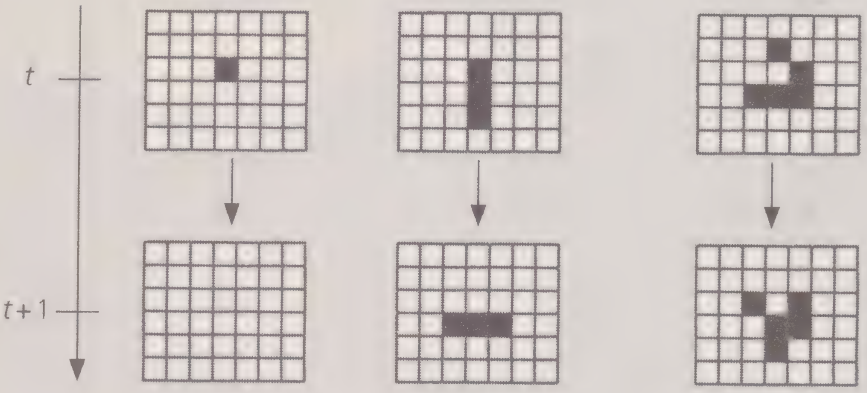
CONWAY'S AUTOMATISM OR THE GAME OF LIFE

The Game of Life, devised by John Horton Conway (b. 1937), is a cellular automaton which, despite its simplicity, gives rise to a fascinating emergent behaviour. There are just two rules, and they have to interact with the eight squares adjacent to each other box, and take account of the state of the square in which the cellular automaton is located:

Rule 1. If the colour of the box is 'white' and exactly three neighbours are 'black' then the colour of the box changes to 'black'; otherwise, it remains 'white'.

Rule 2. If the colour of the box is 'black' and two or three neighbours are also 'black', then the box remains 'black'; otherwise, it changes to 'white'.

If readers have a basic knowledge of computer programming, they should execute these simple norms to see what happens live. Otherwise, we will now show some examples of how the game behaves:



This is the emergent form that appears when programming the rules of the Game of Life; it is known as a 'glider', and is formed by the following cyclical sequence. As you can see in the

animal's immune system, or swarm intelligence, which strives to copy the simple, individual behaviour of each member of a colony (a swarm of bees, for example) in order to, in aggregate form, simulate certain apparently intelligent patterns of behaviour.

An animal's immune system is, in some ways, a highly efficient pattern recognition and optimisation system. Given a new problem to solve (an antigen that has entered the body), it designs a solution through an orderly process of trial and error; in biological terms that is an antibody that recognises the antigen in question.

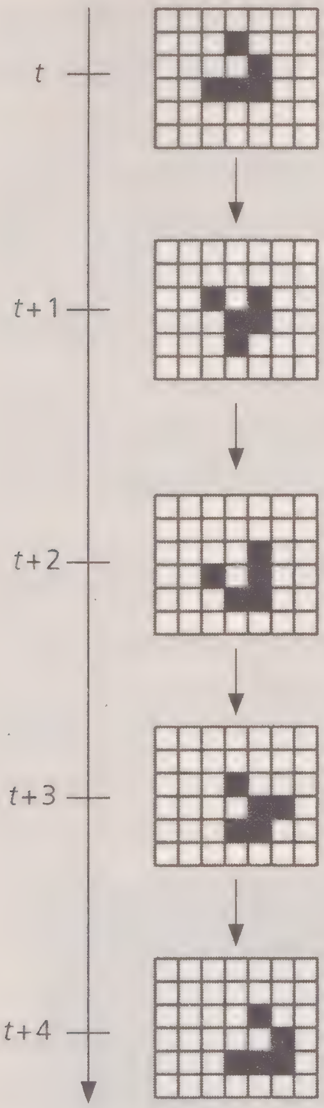
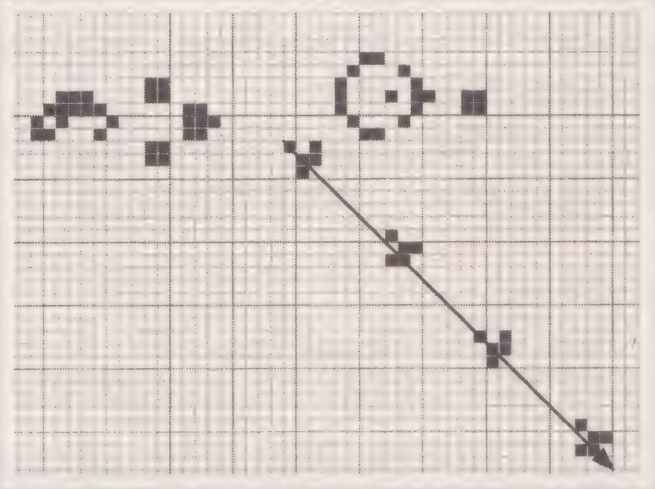


diagram on the left, the shape at $t+4$ is identical to the shape at t, but the whole thing has shifted one square down and one square right. Therefore, if we repeat the operations through to $t+9$, we see that the glider moves across the board diagonally, as shown in the diagram below:



The most sophisticated version of the glider configuration. If the image was animated, we would see how the shapes along the arrow move in the direction it is pointing.

The immune system functions in a similar way to an evolutionary process, with one major exception. In the immune system, the different proposed solutions are not crossed to try to identify a hybrid solution which combines the virtues of its parents. The procedure can be broken down into the following steps:

1. A wide variety of antibodies is randomly generated.
2. The fitness of each antibody is assessed in terms of whether it can recognise the antigen that is attacking the body.
3. A second generation of antibodies is generated using the following strategy:

  a) The antibodies are replicated into multiple copies. Each antibody is replicated multiple times in proportion to its fitness. A highly effective antibody will be replicated many times by the new generation, while an unfit one will either not be replicated or will be replicated only a very few times.
  b) Variations are introduced in the copies of antibodies (or mutations, to use the terminology of evolutionary algorithms), in inverse proportion to their effectiveness. The copies of the effective antibodies will be barely modified in the new population (although some will be), while the replicas of unfit antibodies will undergo major variations.

4. The way in which the new antibodies designed in previous steps cope with the antigen is again evaluated, and the process is repeated to create a new generation of antibodies.
5. When the biological system thinks it has an effective antibody that can recognise the antigen, the process stops.

The process the immune system uses to design antibodies can be adapted easily to solve real problems. In fact, the only critical consideration is how to represent the possible solutions to a problem so that they can be replicated and varied. The recommended course of action in this case is the methodology used in evolutionary algorithms, i.e. codifying the solution using composite chromosomes made up of genes.

Although we are mixing the terminology used in two apparently very different techniques, the invention works and artificial immune systems are increasingly widely used to solve real-life engineering problems, not only because of their effective optimisation, but also because they adapt very well to the architecture of modern computing and supercomputing models based on grid computing and cloud computing. In these cases, the computing power is distributed across an abstract and diffuse 'cloud' of computers, which individually are very powerful, although the communication links between them are not necessarily very good. As a result, the central control of an immune system can send antibodies for evaluation in the cloud, and the central system can then design the next generation. In this scenario, the lion's share of the computing cost lies in the individual evaluation of antibodies, so their evaluation is outsourced to the cloud while new generations can be created on a rolling basis at low cost by the central system.

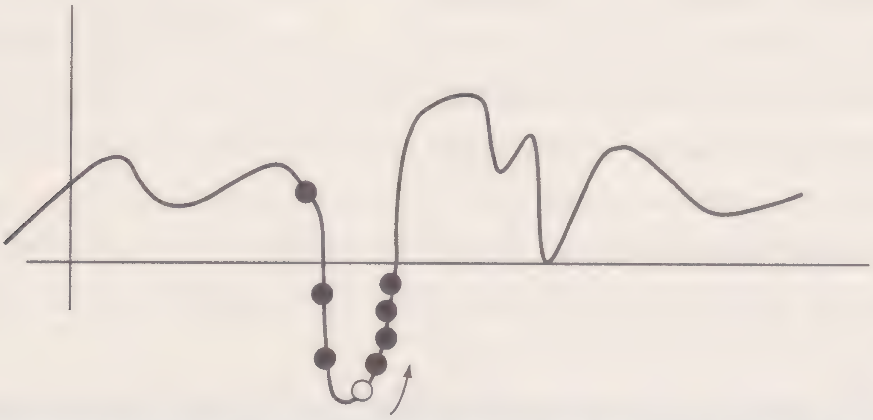
Swarm intelligence

Swarm intelligence is also inspired by nature. The term was coined by Gerardo Beni and Jing Wang in the late 1980s. Swarm intelligence is based on simulating the individual behaviour of single entities so that, when the behaviour of many identical entities is aggregated, an overall behaviour emerges that might have a degree of intelligence. The main challenge in implementing a swarm intelligence system is therefore to define how each entity interacts with its neighbouring entity and with the environment. If this policy is effectively defined, when the activity of all the entities in the colony or swarm is aggregated a globally intelligent form of behaviour should emerge.

Let's consider a practical example inspired by the behaviour of a flock of birds flying through the sky. Imagine we want to optimise a complex mathematical function with hundreds of dimensions, various local maxima and minima, etc. To start with ($t = 0$), we place 100 'birds' randomly, but in clusters, in an area of the function. Whenever we move the time line forwards ($t' = t + 1$), each of them, individually, only has to take account of two things: a) the direction (A) of the 'centre of mass' of the flock, in other words the average direction taken by the

Swarm intelligence is inspired by the way some birds move; starlings, for example, fly together in huge flocks that trace very unusual shapes in the sky.

rest of its companions, so that it doesn't move too far away from them, and b) the direction (*B*) in which the maximum gradient of the function to be optimised is moving; this means that, as we want to maximise the function, we need to know in which direction the function is growing most quickly. Based on the two calculated directions *A* and *B*, we can calculate a third, $C = A + B$, and each 'bird' has to move slightly in this direction *C*. As all the 'birds' are replicating these rules of movement, the flock navigates by the function, without getting too far apart and searching for the global maximum. The advantage of using a group of 'birds' rather than just one is that using various points of interpolation (each of the 'birds') increases the sampled surface area of the function and reduces the chances of falling into local maxima, far from the global maximum.



In the diagram above, the black dots represent the various 'birds' in the flock, and the white dot represents the flock's centre of mass. The arrow indicates the overall direction taken by the flock in search of the global maximum.

However, despite the innovations produced by swarm intelligence, the use of these methods to solve real problems is still at an early stage. There are currently two closely linked spheres in which intensive research is being conducted into these techniques for the automatic control and navigation of vehicles: the aerospace and military sectors.

Applications of artificial life

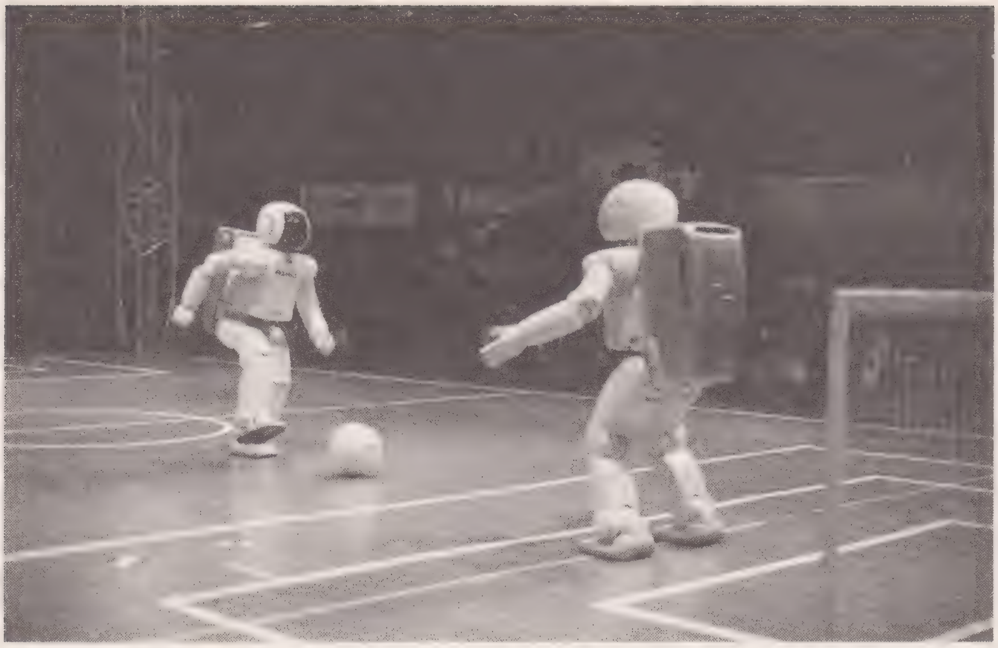
Artificial life is a relatively new area of artificial intelligence. That is why the emerging applications of artificial life are at such an early stage. However, in the future, complex control, supervision and planning tasks will be carried out by 'living' systems, as is already the case with speculative stock market investments.

Game theory

Game theory is a branch of mathematics dedicated to studying the interactions between incentive structures and how to implement decision-making processes. The ultimate aim is to identify optimum strategies and predict the behaviour of the individuals involved in one of these structures in response to a specific situation. The mathematicians John von Neumann and Oskar Morgenstern laid the foundations of this discipline during the Cold War. Their aim was to identify optimum military strategies, although the application of game theory was quickly extended to economics, politics, ethics, philosophy, biology and, of course, computing theory.

Game theory is very useful in the study of complex adaptive systems, as the agents that make up these systems often have to compete or cooperate to promote the system's common good. In a cooperative model, an agent's individual efforts are often greater than the overall benefit spread proportionally among all the agents that make up the system. However, these efforts can be essential for achieving the common good, the benefit of which, in absolute terms, can multiply the individual's efforts by several orders of magnitude. So, to incentivise appropriate behaviour by the agents of the system and be able to predict the viability of a complex adaptive system on the basis of their behaviour, game theory analysis has to be used.

Honda's famous robot, ASIMO, which can perform such human activities as waving, shaking hands and walking down stairs.



Honda's ASIMO robots can even play football.

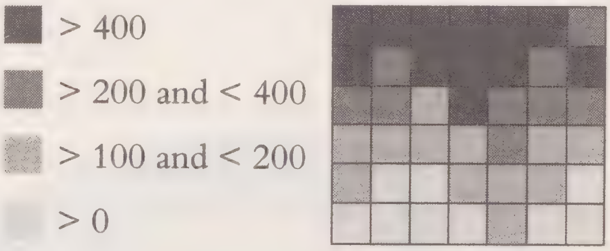
Back to data mining

Artificial life is an evocative concept which is somewhat mysterious to non-specialists. However, the concepts we have discussed so far and which fall within the definition of artificial life, like cellular automata, are used in engineering for fairly prosaic tasks. One of these applications is intelligent data analysis, or data mining, which we have already explored. In a data analysis problem, large quantities of data have to be processed to extract conclusions, but the volumes of information that usually have to be processed are so vast that human beings would find managing them an almost impossible task. That is why intelligent computerised tools are generally used to analyse any trends in the data.

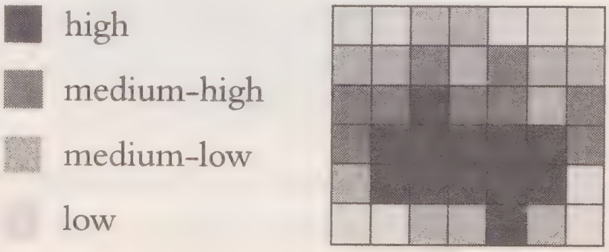
Despite the fact that data analysis can draw on a wide variety of tools, cellular automata contribute something different – the capacity to link data spatially. Imagine we are analysing data on the sales of umbrellas in a particular country. The sales data, broken down by customer, can be processed without taking account of the spatial distribution of sales, or at best, by introducing spatial distribution as a categoric variable, i.e. customer A bought 20 units and A is from city X, while B bought 240 units and comes from city Y, and C bought 4,530 units and is from city Z. In a system that cannot take account of spatial distribution, cities X, Y and Z are merely categories, and it is difficult for it to take account of the fact that city X is 150 km south of city Y, and that Y is 400 km to the south of Z. If this were taken into account, we would see that in this particular country, the northern region gets more rainfall, and therefore the further south you travel the sales of umbrellas fall drastically.

However, if we represented these geographical data on a grid (as cellular automata do) so that the spatial distribution bore a certain relationship to the actual geographical distribution of the data sources, the analysis of this information we could carry out would take account of the spatial distribution more intelligently than a simple categorisation.

For this reason, once the data are distributed on a grid, a methodology like an evolutionary algorithm can be used to identify, through evolution, the rules that the cellular automaton needs to implement to perform the data analysis. Returning to the sales of umbrellas and introducing the sales-linked data item 'rainfall at point of sale', we can ensure that an algorithm evolves a set of rules that paints the cells one colour or another according to the umbrella sales in each point of sale, regardless of the rainfall effect. So, if we coloured the sales map without taking rainfall into account, we would have a map like the following:



However, by eliminating the effect brought about by local rainfall, the map would take on the following appearance:



An expert reading this might infer that most umbrellas are sold in the central-southern region, showing that in this part of the country purchasing power is higher, as, given the characteristics of the region, the population is able to spend its money on a product that is not completely necessary. What the umbrella retailer would then do is to increase the price of umbrellas in the central-southern region of the country because, although fewer units are sold, people buy them as a luxury rather than a necessity and, therefore, are less price-sensitive.

Robot programming

Another area in which artificial life and, more generally, complex adaptive systems, are very important, is the programming of robot behaviour. An increasing number of domestic robots are reaching the mass market; they are capable of carrying out simple cleaning tasks, like vacuum cleaning or mopping the floor, or detecting the presence of intruders. These robots usually have a degree of mobility, as they move around the home using wheels, but have to be guided by an intelligent system to identify which direction they need to move in and which action they need to perform in each location. In the most simple case, that of robot vacuum cleaners, we will examine how they satisfy the definitions of complex adaptive systems:

The vacuum cleaning robot is one of the best-known domestic robots.

— Aggregation. Of course, these robots are aggregate systems, as they have motors, presence detectors, the vacuum cleaner itself, the processor that determines which direction to move in, etc.

— Tagging. These robots can tag and interact with a tagged environment. For example, if they detect that more dirt than normal accumulates in a particular area of space, they tag it as such and focus greater efforts on it. The user can also tag an area where the robot must not go, and the robot can detect it and avoid it.

— Non-linearity. Once again, these robots' behaviour is clearly non-linear, as the whole unit is capable of performing higher-value tasks than the sum of its parts. A motor, wheels, a vacuum cleaner, etc.: separately, these parts could not keep a home dust-free without human intervention, but they do have this ability when they are coordinated within the complex adaptive system formed by one of these robots.

— Flows. The robot is a complex information flow management system: the information flows between the environment and the different parts of the robot. The robot has a certain number of sensors that provide it with data on the environment, such as where the walls are or if it is entering a particularly dusty area. This information flows to the central processor, where it is analysed and further signals are sent to the robot's various motors, which execute the orders that modify the original environment from which the initial signals flowed. If it finds dirt, it will order an increase in suction power, and if it encounters an obstacle, it can order an about turn.

— Diversity. They also display diversity in the way they behave. If they encounter an obstacle, they look for a way to get around it. This behaviour is diverse in that they don't always negotiate obstacles in the same way; instead they alternate methods of avoiding obstacles so as to minimise the chances of getting trapped for ever in the same place.

— Internal models. They boast a number of tacit internal models, and when they start operating they follow an arbitrary route, but as they become more familiar with an area by exploring more surface area, they focus their efforts on the areas where most dust accumulates.

— Building blocks. Finally, these robots use building blocks in their internal models. For example, if they come across a wall, they try to avoid the obstacle, until they realise that it is a wall and the obstacle is unavoidable; they then adopt a different strategy. Naturally, robot vacuum cleaners are not trained to adopt this behaviour when faced with all types of walls or objects that resemble walls; instead the 'wall' building block is used, and when one appears, the strategy is deployed.

Epilogue

Prediction is difficult, especially about the future.
Niels Bohr

Having come this far, readers should now have some idea of what is real and what is invented in all those science fiction stories, novels and films that all too often portray thinking machines with almost human characteristics.

Artificial intelligence is currently no more than a set of advanced tools we can use to solve complex problems quickly and roughly on the basis of acquired experience, as we would as humans. We sometimes have the chance to enjoy a measure of the creativity and innovation made possible by the algorithms that modern science and technology have brought within our reach, but unfortunately we are still far from developing machines, even in the most cutting-edge fields of research, that might be confused with human beings. Nonetheless, on a daily basis, and perhaps without realising it, we interact with tools, systems and devices that really do deploy intelligent algorithms, including many driver-free trains, waste water treatment in most large cities, and the logistical planning systems used to move the thousands of tonnes of merchandise that criss-cross our planet every day.

In fact, little-by-little, artificial intelligence is becoming a part of our lives, and sooner or later the day will come when we can interact with machines with the same level of creativity, perception and emotional intelligence as a human being; then we really will know that we are not alone.

Appendix

Conversation with Eliza

In Chapter 1 we had a short conversation with Eliza, the 'automatic psychotherapist' developed by Joseph Weizenbaum. For this appendix I have tried to hold another, slightly longer, real conversation with her so that readers can see how she behaves in a more extended example. Of course, the reasons given to Eliza are completely false and serve only to illustrate her behaviour.

Eliza: Hello. How are you? Tell me your problem.
Me : I'm stressed.
Eliza: How long have you been stressed?
Me : Since my editor told me that I had to write this book more quickly.
Eliza: And why did your editor tell you to write this book quickly?
Me : Because he wants me to deliver it before January.
Eliza: And is that a real reason?
Me : I think he's under pressure too.
Eliza: Can you explain that a bit further?
Me : He said they needed quite a lot of titles for immediate publication.
Eliza: And does that explain why he wants you to write this book quickly?
Me : What do you think?
Eliza: Why are you asking me?
Me : Because it's obvious.
Eliza: And what other reasons come to mind?
Me : I think other writers are falling behind.
Eliza: Is there some reason why you say that other writers are falling behind?
Me : Because that's what he told me.
Eliza: Do you think that explains anything else?
Me No, just that he's under pressure too.

Eliza: You're being a little negative.
Me: Why do you say that?
Eliza: Are you interested in this subject?
Me: Yes, I'd like to know why you think I'm being a bit negative.
Eliza: Now I see.

At this point the author tired of lying to Eliza and halted the conversation, but readers will have noticed that her behaviour is in reality quite simple. The first thing that Eliza tries to do before any statement is to ask for reasons, then for the reasons for the reasons, and so on. When the conversation heads down a dead end, she tries to get out of it evasively using phrases like "You're being a little negative" or "Now I see".

Glossary

Automaton. Machine, which does not necessarily have to have a physical manifestation, that can be programmed to process input data in specific ways.

Bayesian network. Mathematical tool based on the probabilistic relationships between certain events, using the rules of conditional probability described in the 18th century by the Reverend Thomas Bayes. Underpinning a Bayesian network is the idea that there are chains of events that usually happen and can share probabilities with other chains of events; that is why they are called 'networks', because they are chains of intertwined probabilities.

Boolean logic. This is a form of mathematical logic based on Boolean algebra, where the variables can only possess the values 'true' or 'false'. All modern digital electronics is based on Boolean logic, with the exception of the latest breakthroughs in quantum computing.

Cellular automaton. Specific type of programmable automaton, and the simplest example of artificial life. A cellular automaton behaves in a particular way. It takes the input data from adjacent cells and, depending on their states, adopts one behaviour or another.

Clustering. A technique that consists of assigning statistical objects that meet various criteria into groups. The challenge with clustering tools is to detect the clustering criteria intelligently. Clustering has a range of applications in all scientific disciplines.

Data mining. Branch of data analysis capable of extracting new knowledge and inferring non-evident rules from a large volume of data. Data mining can establish relationships between data of which there is too great a volume for the human mind to process and make conclusive hypotheses.

Decision tree. Computing tool used to classify statistical data. Classification is based on an analysis of the most decisive or discriminating components that classify a data point as belonging to one class or another. They are very simple yet very effective pattern recognition tools.

Diversity. Concept studied in evolutionary computation to determine the genetic variance of the population (set of proposed solutions) of an evolutionary algorithm and how it evolves over time. The study of genetic diversity in an evolution is crucial in determining the optimum configuration of the algorithm and ensuring it doesn't fall into local sub-optima.

Evolutionary algorithm. Search and optimisation method inspired by natural evolution. In an evolutionary algorithm, various possible solutions to a problem are proposed and evaluated, and the best ones are played off against each other to find the optimum solution.

Evolutionary computation. Discipline which studies evolutionary algorithms, their optimum configuration and how they may be applied to problem solving. See evolutionary algorithm.

Expert system. Old intelligent method that entailed creating computer programs that are expert in a particular technical or scientific discipline. These programs' reasoning was strictly limited to the knowledge introduced when they were programmed, and they found it hard to learn from new experiences; for this reason they fell out of use.

Genetic algorithm. Specific class of evolutionary algorithm. Generally speaking, in genetic algorithms, solutions to a particular problem are encoded as an array of bits. The arrays (called genes) that offer the best solutions (or individuals) are crossed and mutated, simulating as far as possible the biological process of evolution. The genetic algorithm was one of the first evolutionary heuristics popularised by intelligent techniques.

Latent variable. Statistical variable which describes various conditions of a data point simultaneously. Some examples of widely used latent variables include the 'richness' of a society or the well-being of a population. These variables bring greater information density by condensing several simple variables into one. There are automatic methods for creating latent variables, such as variable component analysis, which not only create them but also select those which account for the greatest possible variability between data points.

Neural network. Mathematical tool consisting of a network of artificial neurons that can be trained to solve classification problems. Neural networks imitate the behaviour of an animal's nervous system, which is also made up of neurons trained in a learning process.

Overtraining. Takes place when a classification algorithm has been trained to the point that it is incapable of generalising; instead it can only memorise. When this happens the algorithm cannot correctly classify new data, as it can only classify data memorised during the training process. Overtraining usually happens when algorithms are subjected to overly long learning processes.

Principal component analysis. Normally used in its abbreviated form, PCA is a popular statistical tool used to determine the components, or variables, that account for the most variability in the data being analysed.

Shannon's entropy. Mathematical concept commonly used in telecommunications to determine the 'disorder' or entropy of a signal. It is basically a measurement of the number of different symbols and the frequency with which they appear in a signal or data source. Shannon's entropy is also used in cryptography and data compression.

Support vector machine. Powerful and widely used mathematical tool invented by scientist Vladimir Vapnik in the early 21st century and capable of classifying statistical data by adding new 'artificial' dimensions to the variables in a problem. The tool is so-named because, in order to classify statistical data, the known statistical classification vectors or objects are selected to provide a supporting hyperplane that represents the largest separation between the data points in each class.

Swarm intelligence. Complex system of artificial life used to solve particular problems. Swarm intelligence aims to program automata with a very simple form of 'intelligence'. By aggregating tens or hundreds of automata, the global intelligence level increases non-linearly to reach a significant level of group intelligence.

Turing machine. Specific class of programmable automaton that takes its input from an infinite tape and is capable of moving left and right and writing to the tape. It is assumed that a Turing machine is a universal computer, although this has not been mathematically demonstrated. A Turing machine is a theoretical mathematical construct which is very widely used in computing theory to check that new programming languages can simulate the logic of any algorithm, by implementing a Turing machine using said language.

Universal computer. Tool that can simulate the logic of any algorithm.

A universal computer is a theoretical mathematical construct used to check that a new programming language or new electronic device can execute all the functions it is designed to execute.

Bibliography

GOLDBERG, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Boston, Addison-Wesley, 1989.

—: *The Design of Innovation. Lessons from and for Competent Genetic Algorithms*, Norwell, Kluwer Academic Publishers, 2002.

HOLLAND, J.H., *Adaptation in Natural and Artificial Systems*, Cambridge/London, MIT Press/Bradford Books, 1992.

—: *Emergence. From Chaos to Order,* Cambridge, Perseus Books, 1998.

—: *Hidden Order. How Adaptation Builds Complexity*, Reading, Perseus Books, 1995.

MCELREATH, R., Robert, B., *Mathematical Models of Social Evolution. A Guide for the Perplexed*, Chicago, The University of Chicago Press, 2007.

Index

aggregation 116, 137
algorithms
  evolutionary 10, 39, 43, 46-49, 92, 143
  genetic 42, 45-47, 143
artificial neurone 65-66
automaton 126, 143, 144
  cellular 125-128, 134, 135, 143

back-propagation 70, 73
back-tracking 87, 88
Beni, Gerardo 131
Boole, George 29, 30-31, 144
Boolean algebra 30-31
Boolean logic 31-32, 87, 144
branch-and-bound 87-88
building blocks 124-128, 138

case based reasoning 57
chess 16, 17, 20, 28, 35, 124
Chinese room 16-18, 111
closed world assumption 34
clustering 106, 143
constraints 18, 25, 41
Conway's automatism 127, 128
Conway, John 127-129
curse of dimensionality 99-104
CWA 34-35

data mining 95-99, 145
data visualisation 104-106
decision tree 75, 77, 143
Deep Blue 16-17
De Morgan, Augustus 29
diversity 41, 44, 121-123, 137, 144
docking 51

Eliza 32-33, 141-142
evolutionary computation 38-39, 50, 143, 144

feature selection 100
feed forward 69
flash crash 55-56
flows 119-121, 123, 137

Game of Life 127-129
game theory 133-134
glider 128-129
Go 20-21
greedy backward elimination 102
greedy forward selection 102

heuristic 10-11, 27-30, 88
high frequency traders 56
Holland, John 39, 111
Hughes effect 99

instance 117
internal models 123-125, 138
inverse kinematics 72-73

Kant, Immanuel 105
Kasparov, Gary 16-17

latent variable 103-104, 106, 146
linear non-separability 68-69
Logic Theorist 26-27
Lotka-Volterra equations 118-119

mammogram 56-60
map
  fuzzy cognitive 92
  Kohonen 72
  self-organising 72
McCorduck, Pamela 26
McCulloch, Warren 65
Microsoft Research 62, 110
Morgenstern, Oksar 133

network
  Bayesian 62, 145
  Hopfield 71
  neural 9, 65, 69-74, 92, 97, 103, 145
Newel, Alan 26, 28
no free lunch theorem 88-89
non-linearity 117-119, 137

online marketing 61-64
organ transplants 81-84
overtraining 70, 73, 74, 145

perceptron 65-69
pigeon ranking 83
Pitts, Walter 65
Playfair, William 105
Priestley, Joseph 104
principal component analysis 66, 100, 101, 107, 143

random forest 79
Reiter, Raymond 34
robotics 64, 75
Rosenblatt, Frank 65

S-300, missile crisis 90-91
Searle, John 16
Shannon's entropy 77, 144
Simon, Herbert 26, 28
support vector machines 74, 107
swarm intelligence
swarm intelligence 128, 131-133, 144
system
  complex adaptive 114-116, 120-125, 133, 137
  expert 18, 23, 145
  Michigan-type classifier system 107
  Pittsburgh-type classifier system 107
  truth maintenance *see* TMS
systems
  artificial immune 128-131
  symbolic 26-28

tagging 116-117, 120, 137
Threshold Logic Unit 65
TMS 35
Turing, Alan 13, 15, 30
  machine 15, 126-127, 144-145
  test 13-19, 111

Ulam, Stanislaw 125
universal computer 15, 144, 145

Van Langren, Michael 104
Vapnik, Vladimir 74, 144
variable selection 100, 102, 103
Vaucason, Jacques 112
Von Neumann, John 125, 133
Wang, Jing 131
Weizenbaum, Joseph 32, 141

Minds, Machines and Mathematics

Artificial intelligence

We are used to seeing weird and wonderful visions of the future on TV and in movies, where machines have become autonomous and are able to make their own decisions, often with terrifying consequences for humanity. But how can we separate reality from myth? Exactly how far advanced is artificial intelligence today? This book promises a thrilling journey through the future of intelligence and the role that will undoubtedly be played by mathematics in the scientific adventure ahead.